CSP Gateway Configuration Guide

Version 2017.2
2020-06-25

InterSystems Corporation 1 Memorial Drive Cambridge MA 02142 www.intersystems.com

CSP Gateway Configuration Guide
Caché Version 2017.2 2020-06-25

For Support questions about any InterSystems products, contact:

InterSystems Worldwide Response Center (WRC)
Tel: +1-617-621-0700
Tel: +44 (0) 844 854 2917
Email: support@InterSystems.com

Table of Contents

About this Book

This book describes how to manually set up a web server and the CSP Gateway to connect to Caché on supported operating systems.

This book contains the following chapters. See the Table of Contents for appendices.

- Introduction to the CSP Gateway
- Web Servers for Microsoft Windows
- Web Servers for UNIX®, Linux, and Mac OS X
- CSP Gateway Operation and Configuration
- The detailed Table of Contents

1

Introduction to the CSP Gateway

The CSP Gateway provides the communications layer between the hosting web server and Caché when you call a Caché Server Page.

1.1 Who Should Read this Document

The Caché installation includes scripts that perform Web Server and CSP Gateway configuration for common Web Servers and operating systems.

In most cases, installing Caché according to the usual Caché instructions and installing a typical, supported web server provides a system that works with the CSP Gateway without the need to consult this document.

However, if you have an atypical web server architecture or you are an advanced user who wants to get the best out of your environment, you might want to use this document. This document describes the details of procedures for configuring a web server and the CSP Gateway to connect to Caché. It also describes how to use services that the CSP Gateway provides.

1.2 CSP Documentation

Documentation on Caché Server Pages can be found in the following places:

- *Using Caché Server Pages* — describes how to create CSP pages
- *CSP HTML Tag Reference* — a reference to all CSP tags
- *Caché Server Pages Quick Start Tutorial* — gets you started
- *CSP Web Applications Tutorial* — an in-depth tutorial
- *CSP Web Gateway Documentation* — online help on configuring the CSP Gateway is available in this book in the CSP Gateway Management page in the Management Portal **System Administration** > **Configuration** > **CSP Gateway Management**.
- *CSP Gateway Configuration Guide* — when you install Caché, the CSP Gateway is installed automatically and works for most sites. If you need to configure the CSP Gateway manually, use this document. The following two topics in this guide contain procedures on configuring the CSP Gateway:
  - *Configuring the Web Server and the CSP Gateway*
  - *CSP Gateway Operation and Configuration*

The other chapters contain information relating to particular web servers and are outlined in the following section.

1.3 Conventions Used in this Document

The default installation directory for Caché is documented in the Default Caché Installation Directory section of the *Caché Installation Guide*. This guide refers to the default using the variable *install-dir*. Examples may use C:\cachesys or C:\cache-install-dir\ as the installation directory.

Note: When following steps in this document, change paths to match your own installation.

Lines terminated with a back-slash (\) are continued to the next line.

For example, enter the following line, as shown in this document:

```
Init fn=load-modules shlib=CSPn3.dll \
         funcs=csp_term
```

As:

```
Init fn=load-modules shlib=CSPn3.dll funcs=csp_term
```

1.3.1 Gateway components and physical installation paths

Later sections in this guide describe how the Gateway components should be configured with all supported web servers. The installation paths for components should be regarded as examples rather than taken literally. Also, the InterSystems installers create and maintain separate Gateway installations for the internal (private) web server and any third-party web server that might be present on the same host. In this context 'third-party web server' refers to a web server that is not part of the software installed by InterSystems.

The precise installation location of Gateway components is not particularly critical provided:

- The physical installation paths match those given in the hosting web server configuration where appropriate.
- The security settings, in relation to required access for individual components, are adjusted appropriately. This is particularly important for Gateway components that are accessed directly by the web server since web servers are usually locked down to the extent that the files they are able to access (and executables that can be run) are carefully controlled. It should be borne in mind that security considerations will also be important for any Gateway configuration (and log) files that are accessed by Gateway binaries that are themselves bound to the web server core executable.
- The security policy of the hosting web server is respected. Some web servers – notably that shipped with Secure Linux (SELinux) – are configured such that it is not possible for them to access files that lie outside their own file system. This restriction will clearly have an impact on where certain web-server-facing Gateway components can be installed.

There are four types of Gateway component to consider.

1. Binaries to be loaded by the web server (API based extensions).

   This includes Windows DLLs and UNIX Shared Objects:

   ```
   CSPms[Sys].dll
   CSPn3[Sys].(dll|so|exe)
   CSPa*[Sys].(dll|so)
   mod_csp*.(so|exe)
   ```

   The physical location where these are installed should match the corresponding configuration directives in the hosting web server configuration. This includes directives indicating which third-party modules should be loaded. The web

server requires permission to read and load these modules. Modules named CSP* require permission to read and write to the Gateway configuration and log files (CSP.ini, CSP.log). These are usually created in the same location as the Gateway binaries.

When considering access control for these modules, bear in mind that it is the web server *worker processes* that need to be able to access the modules together with any dependent configuration and log files. For example, in the case of Apache, the server is usually started with superuser permissions but the worker processes that actually serve web requests run with a much lower level of authority (as indicated by the User and Group directives in the Apache configuration file). It is the User and Group specified for the worker processes that should be granted permission to load the Gateway modules and (where appropriate) the ability to read and write to the configuration and log files (CSP.ini and CSP.log).

2. Executables to be called by the web server (CGI modules). (Not all configurations require these executables.)

   ```
   [nph-]CSPcgi[Sys][.exe]
   ```

   The physical location where these are installed should match the corresponding configuration directives in the hosting web server configuration. This will include directives indicating which web requests should be processed by these CGI modules.

   The worker processes of the hosting web server require execute permission for these modules. There are no further dependencies.

3. Static files to be returned by the web server.

   Note: With current CSP Gateway configurations, CSP is often configured to serve static files directly from Cache as opposed to having the web server return them. This section does not apply to such configurations.

   JavaScript modules (such as CSPBroker.js, CSPxmlhttp.js, and so on)

   Java applets (such as CSPBroker.class, CSPBroker.jar, and so on)

   Images (such as created-with-csp.gif, and so on)

   The worker processes of the hosting web server require Read permissions for these files.

4. The CSP Network Service Daemon (NSD)

   Note: Not all configurations require this facility.

   ```
   CSPnsd[Sv][.exe]
   ```

   This can be installed anywhere and the web server does not need to be aware of its physical location since communication between these two points is over TCP (usually port 7038).

   The NSD requires permission to read and write to the Gateway configuration and log files (CSP.ini and CSP.log), which are usually created in the same location.

   Note: For security reasons, do *not* install this module in a location that is accessible by the web server. This module should not share a location with the modules listed in steps 1, 2 or 3. Many web server configurations described in this document explicitly exclude this module from the list of accessible files that can be accessed by the web server. However, it is much safer to physically install the NSD elsewhere in the file system.

Gateway Cache and Permanent Storage

The Gateway, where possible, locates the content of large files (such as large JavaScript files in Zen applications) together with the accompanying HTTP response headers in permanent storage. The essential control information (expiry time and so on) and the index of all cached files is in the shared memory sector. With this architecture, each cache entry consumes a small amount of cache space, in terms of memory usage, quite possibly no more than one cache block per entry.

Cached content is stored in files of type .dat in the Gateway's temp directory, placed by the install script directly beneath the Gateway's installation directory. For example, in a typical IIS installation this is in: C:\inetpub\CSPGateway\temp. The location needs full read/write/delete permissions for the hosting web server worker processes.

1.4 Supported Web Servers

More detailed information on supported web servers can be found in the section "Supported Web Servers" in the online *InterSystems Supported Platforms* document for this release. The following table summarizes the web servers discussed in this document.

Operating System	Web Servers
Microsoft Windows	Microsoft — Internet Information Services (IIS)
	Apache
	Nginx
UNIX®	Apache
	Sun — Sun Java System (on Oracle Solaris only)
	Nginx

The CSP Gateway provides high-performance connectivity solutions for Microsoft, Oracle (formerly Sun and Netscape), Apache, and Nginx web servers. In addition to these solutions, connectivity to Caché through the Common Gateway Interface (CGI) is available for all supported Operating Systems.

Both the Microsoft and the Oracle line of web servers support a multithreaded API which allows extensions, in the form of dynamically bound libraries, to be made to the web server's core functionality. Current versions of the CSP Gateway make full use of these APIs in order to bring high-performance web connectivity to the Caché system. The Windows version of Apache also operates in an exclusively multithreaded mode and, as such, can also take advantage of the CSP Gateway implemented as a dynamically bound library.

The UNIX® versions of Apache are architecturally different from the Microsoft Windows based web servers in that they are not exclusively multithreaded. Apache version 1.3 under UNIX does not use threads under any circumstances and operates exclusively in accordance with the traditional multi-process server model. Apache version 2 (and later) is implemented using a hybrid model made up of threads and multiple processes. In this model, each UNIX process is effectively a multithreaded server in its own right. The Oracle web server can operate as either an exclusively multithreaded server or as a hybrid multithreaded and multi-process server.

The Apache web server publishes a proprietary API in addition to supporting extensions implemented as CGI modules. Extra functionality can be added to Apache by means of user-defined modules (compiled C programs). In fact, a large part of Apache's core functionality is implemented as a set of modules. You can add modules to Apache by one of two methods. First, the source to the module can be compiled directly into the Apache core. This option arguably offers the best performance but, unfortunately, involves reconfiguring and rebuilding the web server. As an alternative to building the module source directly into the Apache core, current versions of Apache (1.3 onwards) support extensions implemented as dynamically linked libraries. This facility allows you to take advantage of the high performance of Apache modules without the need to physically build the module into the core of Apache. The CSP module is distributed as a Windows Dynamic Link Library (DLL), and as a UNIX® Dynamic Shared Object (DSO). UNIX® Shared Objects are conceptually similar to a Windows Dynamic Link Library (DLL) and are linked at run time. The overhead involved in linking to a library at run time is very low on modern operating systems.

A more recent addition to the set of web servers supported by CSP is Nginx. Unlike other web servers, Nginx is based on an asynchronous event-driven architecture. With the event-driven architecture, notifications or signals are used to mark the initiation and completion of each individual operation. A consequence of this design is that while web requests are being

processed, resources can be temporarily released and used by other operations. Resources can be allocated and released dynamically and are only associated with the processing of a web request while they actually required. This leads to a highly optimized use of memory and CPU. The asynchronous nature of this architecture results in threads executing concurrently without blocking each other, thus further enhancing the sharing of resources that might otherwise be associated with a thread waiting on a blocking operation. Nginx is supplied with an API to allow extensions, such as CSP, to be added to its core functionality. However, unlike other web servers, extension modules must be built into the web server core at compilation time. Nginx does not support dynamically loaded extension modules.

All supported web servers can be extended to support CSP through modules that are dynamically loaded by the hosting web server. It is expected that most CSP installations will take advantage of these high-performance web connectivity solutions. An alternative architecture in which the functionality of the CSP Gateway is implemented as a stand-alone executable, operating in its own process and not directly connected to a web server is also provided. This version of the CSP Gateway is known as the *Network Service Daemon* (NSD). In this context, the NSD is responsible for providing the CSP Gateway's core functionality and maintaining persistent connections to Caché. The web server communicates with the web server via small modules of which there are two types: modules that work to the hosting web server's proprietary API and modules implemented as CGI executables. The NSD-based architecture is therefore used in cases where there is either a requirement to either extend the web server by means of the CGI standard or in cases where it is desirable to disengage the functionality of the CSP Gateway from that of the hosting web server.

1.5 Configuring the Web Server and the CSP Gateway

The Caché installation performs web server and CSP Gateway configuration for common web servers and operating systems.

After installing Caché and the CSP Gateway, consult the sections in this book relevant to your system to map file extensions for your system. The appendices in this book have configuration information for atypical CSP Gateway configurations.

To install the CSP Gateway on a remote server (that is, a system that is not running an instance of Caché), you can use one of two methods. On the remote server, you can

- Run the Caché installation script and, on the **Setup Type** page, select **Web Server** only or
- On UNIX platforms only, you can run the standalone CSPGateway installation script. The script asks for information about the remote Caché server: name, address, port, and optional password. The script automatically configures csp.ini based on this information.

For more information on configuring a remote web server, see the section Using Caché Server Pages with a Remote Web Server in this book. After installing the CSP Gateway, consult the sections in this book to map file extensions for your system.

Note: To prevent runtime errors, for High Availability configurations running over CSP, InterSystems recommends that you use a hardware load balancer with sticky session support enabled. For more information, see the section "CSP Gateway Considerations" in the Caché High Availability Guide.

1.5.1 Configuring the CSP Gateway for Ensemble

The CSP Gateway provides web connectivity for both the Caché and Ensemble product lines. The Gateway components are the same in both cases. A Gateway installation provided with a Caché installation can provide web connectivity to an Ensemble installation and vice versa.

For simplicity, this document is written for use with Caché installations. However, with the exception of default path names, the instructions contained in this document apply equally to Ensemble. The default Ensemble names replace `Cache` with `Ensemble`.

1.5.2 Gateway Management Module configuration

Gateway architectures that work directly to a hosting web server's API typically consist of two modules: A Management Module (for example, CSPmsSys.dll) and a runtime module (for example, CSPms.dll). The runtime Module is responsible for processing requests for CSP files and the Management Module provides the Gateway's Management interface. In the CSP Gateway, the runtime Module assumes responsibility for loading and routing management requests to the management module. All requests for the CSP Gateway (CSP and management) are processed by the runtime Module. The Management Module must be installed in the same location as the runtime Module

1.5.3 File Types Served by CSP

Files of type .csp, .cls and .zen are processed in Caché by CSP. All other files (static files) can be served by the web server or CSP. CSP can serve any type of file that is placed in the CSP applications path (including static files). Setting up CSP to serve static files simplifies the web server configuration for CSP applications because you, thus, do not need to create aliases in the web server configuration to represent the locations where an application's static files are held. Setting up CSP to serve static files resolves issues of contention when a single (that is, common) web server serves two different versions of Caché, each requiring different versions of certain static files (for example, hyperevent broker components).

To have CSP serve static files for a particular CSP application, place the static files in the CSP application's file system in the correct location relative to the CSP files that make up the application (not in the web server's own documents file system). (Note that if you are serving files containing Unicode text, CSP uses the BOM to determine the correct encoding to use. The BOM must be present in Unicode text files.)

Consult the sections in this book for your platform.

Note: To run Zen-based applications, you must enable the **Serve Files** option and properly configure your web server. See the section Static Files in *Using CSP* for more information.

1.5.4 Serving Static Files from Caché

You can configure web servers and Gateway installations so that Caché assumes responsibility for serving static files. The Management Portal and the Zen samples are supplied configured for Caché to serve all components in the application. However, it is still possible to configure the web server so that it retains responsibility for serving statics.

1.5.5 Hybrid Multi-Process/Multi-Threaded Web Server Architecture

The Gateway contains enhanced support for the hybrid multi-process/multi-threaded web server architecture. Apache version 2.x.x under UNIX is an example of a web server implemented according to this architecture.

With earlier builds of the Gateway, the stand-alone modules (CSPa2*[Sys].so) were supported but each and every hosting web server worker process would bind to and create an independent instance of the Gateway. In other words, each worker process would load its own instance of the Gateway which would, in turn, load its own copy of the running configuration, manage its own connection table and form cache. The main observable symptom of this behavior being the variability in the Gateway System Status form each time **Refresh** was selected. Each refresh of the Status form would result in the status for just the worker process and Gateway instance that happened to be processing the request for the status form. Also, state-aware mode (preserve mode 1) could not be supported in the multi-process architecture since this mode of operation depends on being able to route all requests for a session to the same Caché process which, in turn, means that all connections to Caché must be held in a single multithreaded process.

An architecture in which all Gateway resources are duplicated across all web server processes is unnecessary and wasteful.

The core Gateway resources are now held in the shared memory sector. All web server worker processes hare a common running configuration, connection table and form cache. The Gateway System Status form shows the status for the whole

web server installation instead of just that of a single worker process. The status form's connection table includes an extra column with the web server process ID with respect to each connection to Caché.

Finally, state-aware sessions are supported in the multi-process architecture. Although the connection pool (to Caché) is distributed amongst several web server processes, the Gateway uses an InterProcess Communications (IPC) protocol to route requests for state-aware sessions to the correct hosting process in the web server environment.

1.5.6 Gateway Registry

The Gateway is supplied with the new Caché based Gateway Registry. All web server and Gateway installations are registered with Caché as they connect. The registry contains the infrastructure to allow Caché code to interact with connected Gateway installations for the purpose of reading and writing the configuration and monitoring the system status and Event Log.

1.5.7 Enable Sticky Sessions on Hardware Load Balancer on High Availability Solutions

For High Availability solutions running over CSP, InterSystems recommends that you use a hardware load balancer for load balancing and failover. InterSystems requires that you enable sticky session support in the load balancer; this guarantees that – once a session has been established between a given instance of the gateway and a given application server – all subsequent requests from that user run on the same pair. This configuration assures that the session ID and server-side session context are always in sync; otherwise, it is possible that a session is created on one server but the next request from that user runs on a different system where the session is not present, which results in runtime errors (especially with hyperevents, which require the session key to decrypt the request). See your load balancer documentation for directions on how to enable sticky session support.

Note: It is possible to configure a system to work without sticky sessions but this requires that the CSP session global be mapped across all systems in the enterprise and can result in significant lock contention so it is not recommended.

For more information on high availability and CSP, see the section "CSP Gateway Considerations" in the Caché High Availability Guide.

1.5.8 Enable Script to Reactivate Gateway Configuration

You can enable an external (non-Cache based) script to reactivate the Gateway's configuration.

Scripts should add the following line (case-sensitive) to the SYSTEM section of the Gateway configuration file:

```
[SYSTEM]
RELOAD=1
```

The Gateway caretaker daemon checks the RELOAD flag approximately every minute and, if correctly set, reloads and reactivate its configuration and removes the flag from the file. The following message is written to the Event Log after a successful reload operation:

```
Gateway Management
Gateway Configuration Reloaded and Reactivated
```

1.6 Private Web Server and Management Portal

The Management Portal Apache server is self-contained and configured to listen on a non-standard TCP port (something other than the usual, well known, HTTP server port of 80). It does not interfere with any other web server installation operating on the same host.

To access the Management Portal, enter the following URL, which resolves to the port number on your private web server for the current Caché instance:

http://localhost:57772/csp/sys/UtilHome.csp

The minimal Apache server used for the Management Portal is often referred to as the Private Web Server.

A minimal build of the Apache web server is supplied for the purpose of running the Management Portal. This server is known as the Private Web Server (PWS) and is built and configured to meet the management needs of InterSystems server products and is configured to only connect to Cache and Ensemble. The options selected to create the PWS are not, in general, suitable for production use. In particular, security is minimal and the configuration deployed is generally unsuitable for applications for which a high volume of HTTP requests is anticipated. Testing (by InterSystems) of the PWS only covers the use of this server for managing Caché and Ensemble. However many developers find it useful to use the PWS for testing their own CSP and Zen applications.

Note: When installing Caché and Ensemble, this private version of Apache is installed to ensure that:

1. The Management Portal runs out of the box.
2. An out-of-the-box testing capability is provided for development environments.

The private Apache web server is not supported for any other purpose.

For deployments of http-based applications, including CSP, Zen, and SOAP over http or https, you should not use the private web server for any application other than the Management Portal; instead, you must install and deploy one of the supported web servers (for information, see the section "Supported Web Servers" in the online *InterSystems Supported Platforms* document for this release).

The PWS is responsible for supporting the management portals for Caché, Ensemble, and HealthShare Foundation. However, customers are not required to use this web server to manage InterSystems products: customers may run the various management portals through a web server of their own choosing.

Finally, the PWS is self contained and configured to listen on a non-standard TCP port (something other than the usual, well known, HTTP server port of 80). It will not interfere with any other web server installation operating on the same host.

Note: The PWS is available for UNIX, Linux, Mac OS X and Windows. See the section Building the Private Web Server for more information.

The entry point for the Management Portal is normally via the following CSP path and file:

```
/csp/sys/UtilHome.csp
```

For example:

```
http://127.0.0.1:8972/csp/sys/UtilHome.csp
```

1.6.1 Building the Private Web Server

The (default) full Apache server is usually created with the following sequence of commands:

```
./configure --prefix=<install-dir>
make
make install
```

The minimal Apache build is typically created as follows:

```
./configure --prefix=/usr/cachesys/httpd --with-port=57772 \
            --with-pcre=$srcdir/pcre \
            --enable-mods-static="log_config mime alias unixd authz_core" \
            --disable-ssl \
            --enable-so --without-gdbm --without-ndbm \
            --without-berkeley-db --with-included-apr --with-expat=builtin \
            --with-mpm=prefork --disable-shared
make
make install
```

Notice that many of the services that are normally required for a production grade installation are excluded.

While this server can be used to host other CSP applications it is strongly recommended that a full, independent web server installation is used for this purpose. It should be remembered that any changes made to the configuration of the Management Portal Apache installation are overwritten when the hosting Caché installation is upgraded.

The Management Portal Apache installation uses the following CSP Gateway modules for communicating with Caché:

- *Windows*: CSPa2.dll and CSPa2Sys.dll
- *UNIX*®: CSPa2.so and CSPa2Sys.so

1.6.2 Managing the Private Web Server

Under normal operational conditions, the Management Portal Web Server for a particular instance of Caché is started when Caché is started and closed down when Caché is closed down. Occasionally it may be necessary to restart the Management Portal Web Server without disrupting the corresponding Caché server. For example, a web server restart is necessary if a configuration change is made to the web server (httpd.conf).

Use the following commands to start and stop the Management Portal Web Server.

Windows

Start the Management Portal Web Server:

```
<cache-install-dir>\httpd\bin\httpd -k start -n <instname>httpd -c "Listen <port>"
```

Stop the Management Portal Web Server:

```
<cache-install-dir>\httpd\bin\httpd -k stop -n <instname>httpd
```

For example:

Caché installed in: C:\cachesys

Caché instance name: CACHE

TCP port for Apache: 57772

Start:

```
C:\cachesys\httpd\bin\httpd -k start -n CACHEhttpd -c "Listen 57772"
```

Stop:

```
C:\cachesys\httpd\bin\httpd -k stop -n CACHEhttpd
```

UNIX®

Start the Management Portal Web Server:

```
<cache-install-dir>/httpd/bin/httpd -d <cache-install-dir>/httpd -c "Listen <port>"
```

Stop the Management Portal Web Server:

```
kill `cat <cache-install-dir>/httpd/logs/httpd.pid`
```

Note: On AIX®, *LD_LIBRARY_PATH* must include the /cache-install-dir/bin directory in order to manually run httpd.conf.

For example:

Caché installed in: /usr/cachesys

TCP port for Apache: `8972`

Start:

```
/usr/cachesys/httpd/bin/httpd -d /usr/cachesys/httpd -c "Listen 8972"
```

Stop:

```
kill `cat /usr/cachesys/httpd/logs/httpd.pid`
```

Note: On AIX®, *LD_LIBRARY_PATH* must include the /cache-install-dir/bin directory in order to manually run httpd.conf.

1.6.3 Limitations of the Private Web Server

This section discusses the differences between the configuration of the PWS and that of a typical production grade Apache installation.

Windows

Windows-based Apache installations use a special multithreaded form of the Apache Multi-Processing Module (MPM) which is better suited to the way the operating system is optimized. Therefore, the behavior of the PWS under Windows is similar to that of a production grade Apache build as far as the ability to handle concurrent load is concerned.

If high availability and production-grade security is a requirement, or there is a need to integrate with other sources of web information, or a need for a high degree of control over the web server, a separate production-grade build of Apache is recommended - ideally operating on its own server. If, on the other hand, low volumes of HTTP traffic are expected, and there are limited demands for high availability and security, then the PWS may be suitable for deployment under these circumstances.

UNIX

The PWS defaults to using the Apache Group's prefork Multi-Processing Module (MPM). This is a non-threaded server model: the number of requests that can be concurrently served is directly related to the number of Apache worker processes in the pool.

The PWS is configured to occupy the smallest possible footprint by allowing a maximum of two worker processes to be created for the pool. The following settings will be found in the Apache configuration (httpd.conf) for the PWS:

```
MinSpareServers 1
MaxSpareServers 2
```

By contrast, the default Apache configuration for a production grade build is usually as follows:

```
StartServers       5
MinSpareServers    2
MaxSpareServers   20
ServerLimit      256
MaxClients       256
```

This configuration allows Apache to create 5 worker processes at startup, increasing to a maximum of 256 as the concurrent load increases. Because of these differences in configuration, the performance of the PWS is noticeably inferior to that of a production grade Apache build. This performance deficit becomes more noticeable as the concurrent load increases. However, it is possible to change the configuration of the PWS to match that of a full Apache installation (shown above). Apache must be completely restarted after changing these parameters.

2

Web Servers for Microsoft Windows

This section describes how to manually configure Web Servers from Microsoft and Apache on systems running Windows.

When you install Caché, you can select the option to configure your web server to work with CSP. This is what most customers do. If you do not choose this option during Caché installation, you can run the Caché installation again and select to install only the CSP Gateway components. If you do not select to configure your web server during the Caché installation, you can configure your web server manually to work with CSP. To configure your web server manually on a system running Windows, use the instructions in this chapter.

This chapter describes the most common configuration for each of the web servers, IIS 7 (and later versions), IIS 6, Apache. Other, atypical, configuration options are described in the Appendix, Alternative Configurations for Microsoft Windows.

- If you are using an IIS web server, follow the directions in this chapter for your configuration. First follow the steps in the section "Microsoft IIS All Versions" Then follow the section that is applicable, either "Microsoft IIS 7 " or "Microsoft IIS 6 ".
- If you are using an Apache web server, following the directions in this chapter to configure Apache using Native Modules, extensions implemented as dynamically-linked modules (DLLs) and a means through which *ISAPI* extensions, a high-performance API, the Internet Server Application Programming Interface (ISAPI), developed for Microsoft's web servers, can be utilized.

2.1 Microsoft IIS All Versions

If you are using IIS as your web server, follow the steps in this section.

2.1.1 Summary of Configuration

Follow these steps to configure your web server for IIS, all versions.

1. Set permissions for the Gateway components. For details, see the section "Setting Permissions for the Gateway Components".
2. Configure the CSP application path. For details, see the section "Configuring the CSP Application Path".
3. Enable URLs with /bin.

   For details, see the section "Enabling URLs with /bin".

4. Turn to the section for your web server. If you are using IIS 7 or later, see the section "Microsoft IIS 7". If you are using IIS 6, see the section "Microsoft IIS 6". Or, if you have an atypical configuration, see the appendix Alternative Configurations for Microsoft Windows.

2.1.2 Setting Permissions for the Gateway Components

Regardless of which CSP Gateway configuration option you choose, you need to assign appropriate permissions to web resources held outside the standard IIS documents root (for example, C:\inetpub\wwwroot).

IIS 7 does not, by default, allow the user of a web application to access anything outside the scope of the pre-configured documents root unless you assign Read & Execute and Write permissions for those external resources to the following user or groups:

[machine_name]\IIS_IUSRS

And:

[machine_name]\Users

It should be noted that **IIS_IUSRS** represents the user (group) under which IIS worker processes operate. It essentially replaces the more familiar **IUSR_[machine_name]** user group found in earlier versions of IIS. Applications controlled through IIS (such as the CSP Gateway) operate with the level of privilege assigned to **IIS_IUSRS**.

For CSP, resources external the web server's root usually include the following:

Gateway binary components:

C:\inetpub\CSPGateway

Static file components:

cache-install-dir\CSP\

Permissions can be manually assigned to these folders via Windows Explorer as follows:

1. Right select the folder name and select **Properties**.
2. Select the **Security** tab.
3. Select **Edit**.
4. Select **Add**.
5. In the **Enter the object names to** text box enter:

   ```
   [machine_name]\IIS_IUSRS
   ```

6. Select **Check Names** and **OK**.
7. Select **[machine_name]\IIS_IUSRS** in the **Group or Usernames** window, then:
8. Assign **Read & Execute** and **Write** permissions in the **Permissions** window.
9. Select **Apply** and **OK**.
10. Repeat the above process for the **[machine_name]\Users** user group.

As with previous versions of IIS, full read and write permissions for the Gateway configuration and event log files (CSP.ini and CSP.log) should be assigned to the IIS user group. For example, at the Windows command prompt, enter:

```
cacls CSP.ini /E /G IIS_IUSRS:F
```

```
cacls CSP.log /E /G IIS_IUSRS:F
```

Of course, this can also be done via Windows Explorer.

2.1.3 Configuring the CSP Application Path

This section describes the procedure for configuring the CSP application path (such as /csp) for IIS. These procedures are common to all CSP Gateway configuration options for IIS.

As with previous version, IIS is configured in the **Internet Information Services (IIS) Manager** control panel. Subdirectories configured under the documents root can either be classed as **Virtual** or **Applications. Virtual** subdirectories (or aliases) are mapped to physical equivalents (windows directories). The same applies to subdirectories classed as **Applications** except that, in addition to defining the physical equivalent, you can associate the application with a particular application pool (the default of which is **DefaultAppPool**).

Since CSP applications are served through the CSP Gateway, the hosting subdirectories (such as/csp) should be configured as **Applications**.

In a default CSP configuration, the /csp application path is mapped to the physical location *install-dir*\CSP. All the static files are located under this root (\csp\broker…).

1. Open the **Internet Information Services (IIS) Manager**.
2. In the left panel, expand the top level to reveal the **Web Sites** section, then the **Default Web Site** section. Highlight the **Default Web Site** section:

   ```
   [MACHINE_NAME] ([machine_name]\[user_name])
               Web Sites
                       Default Web Site
   ```

3. In the right panel, select **View Applications**.
4. Again in the right panel select **Add Application**.
5. In the **Add Application** dialogue, enter:

   Alias: **csp**

   Physical path: *install-dir*\CSP\
6. Select **OK**.

If you are using a CSP Gateway solution based on an atypical option, set up an application called /bin under the /csp application. Map this to the physical directory holding the Gateway binaries. For example:

Map application /csp/bin to C:\inetpub\CSPGateway

2.1.4 Enabling URLs with /bin

If you installed the CSP Gateway using the Cache installer, this step was done automatically for you. If you are installing the CSP Gateway manually, you need to do this step. (See this external web site for more details and alternative ways to accomplish this http://weblogs.asp.net/owscott/archive/2008/03/05/iis7-blocks-viewing-access-to-files-in-bin-and-other-asp-net-folders.aspx.) To enable URLs that contain /bin, add the following location tag to your applicationHost.config file:

```
<location path="sitename.com/subfolder/bin/debug">
    <system.webServer>
        <security>
           <requestFiltering>
              <hiddenSegments>
                 <remove segment="bin" />
               </hiddenSegments>
           </requestFiltering>
        </security>
     </system.webServer>
 </location>
```

2.1.5 Restarting IIS

This section describes what happens when IIS is restarted via the various control panels:

Most configuration changes can be made in real-time to an active IIS installation. However, the **Internet Information Services (IIS) Manager** control panel provides stop, start, and restart options. These are useful for the refreshing the web server configuration but does not result in an active Gateway installation being reinitialized (the Gateway DLLs are not reloaded).

As with previous versions of IIS, if you want to force IIS to restart, so that the Gateway modules are reloaded, then you have to restart the **World Wide Web Publishing** service via the main Windows **Services** control panel.

2.1.6 Troubleshooting

This section describes problems that commonly occur in configuring third-party modules (both Native and ISAPI) to work with IIS.

The most common problem likely to be encountered is that, after reconfiguring, requests to IIS fail with the following error:

```
Service Unavailable

HTTP Error 503. The service is unavailable.
```

This usually indicates that the default **Application Pool** has terminated.

1. Open the **Internet Information Services (IIS) Manager** window.
2. In the left panel expand the top level to reveal the **Application Pools** section.

   [MACHINE_NAME] ([machine_name]\[user_name])

   Application Pools
3. Check that the Default Application Pool (`DefaultAppPool`), or whatever application pool your server is configured to use, is marked with a Status of **Started**.
4. Restart the application pool if necessary (using the options in the right panel).
5. If problems persist, look for clues in the main **Windows Event Log**: the **Applications** section. In particular, check for the following error message:

   ```
   Failed to find the RegisterModule entrypoint in the module DLL
   C:\inetpub\CSPGateway\CSPms.dll. The data is the error.
   ```

   This, for example, indicates that the version of Gateway DLLs that you are using do not implement the Native Modules interface. Either obtain later DLLs from InterSystems or configure the Gateway to work through the conventional ISAPI interface.

As with all software, restarting often clears transient problems: To completely restart IIS, restart the **World Wide Web Publishing** service via the main Windows Services control panel.

Do not use the `Add Wildcard Script Map` utility to map file extensions. If you do, you may see this error: `The specified module required by the handler is not in the modules list. If you are adding a script map handler mapping, the IsapModule or the CgModule must in the modules list.` Instead use `Add Module Mapping for *` to map file extensions using a wildcard.

If URLs with /bin in them are not working, see the section “Manual Step for Enabling URLs with /bin”

2.2 Microsoft IIS 7 or Later

In this build, the Microsoft ISAPI extensions (CSPms.dll, CSPmsSys.dll and CSPcms.dll) have been adapted such that they can work directly to the Native Modules interface in IIS 7. This is the web server supplied with Windows Vista and Windows Server 2008.

The Gateway modules that we supply can work with the Native Modules in IIS 7. They can also be used with ISAPI extensions. There are additional configuration options for customers who are using the NSD. This section describes how to configure your IIS 7 web server to work with the Native Modules. The appendix Alternative Configurations for Microsoft Windows contains information on configuring IIS 7 for other configurations.

2.2.1 Install Locations for IIS 7

Install the CSP Gateway components and the CSP static files as follows:

1. The default location for the Native Modules

   - CSPms.dll (Runtime module)
   - CSPmsSys.dll (Systems Management module)

   The default location for these modules is:

   C:\Inetpub\CSPGateway

   The configuration file (CSP.INI) and Event Log (CSP.LOG) are written in this directory for non NSD-based connectivity options.

2. HyperEvents Components

   - CSPBroker.js
   - CSPxmlhttp.js

   The default location for these files is:

   C:\cache-install-dir\csp\broker

3. Miscellaneous static resources used by the CSP Samples

   A number of static web resources (such as image files) are required by the CSP Samples. The default location for these files is:

   C:\cache-install-dir\csp\samples

4. Miscellaneous static resources used by the Management Portal (Caché v5.1 and later)

   A number of static web resources (such as image files) are required by the Management Portal. The default location for these files is:

   C:\cache-install-dir\csp\sys

2.2.2 Recommended Option: Using Native Modules (CSPms*.dll)

This is the recommended and most-used configuration option. It uses the new Native Modules interface supplied with IIS 7. This option provides the best performance.

For other configuration options using ISAPI or NSD, see the appendix Alternative Configuration Options for Microsoft Windows.

Register the Native Modules and configure the web server so that it recognizes CSP requests (files of type .csp, .cls, and .zen) and passes them to the CSP Gateway for processing. Include any additional files that might be required for your installation (such as, for example, special CSP resources needed for DeepSee).

2.2.2.1 Registering the Native Modules

DLLs: CSPms.dll and CSPmsSys.dll

Before these modules can be used they must be registered with IIS. This is done in the **Internet Information Services (IIS) Manager** control panel.

1. Open the **Internet Information Services (IIS) Manager** window.
2. In the left panel, highlight: **[MACHINE_NAME] ([machine_name]\[user_name])**
3. In the middle panel, double-click the **Modules** icon.
4. In the right panel, select **Add Native Module** (or **Configure Native Modules**). The precise wording depends on the build of IIS in use.
5. Select **Register** and enter the following in the **Register Native Module** dialogue:

   Name: **CSPms**

   Path: C:\inetpub\CSPGateway\CSPms.dll

   Select **OK**.
6. In the left panel, expand the top level and expand **Web Sites**, and **Default Web Site**. Highlight **Default Web Site**:

```
[MACHINE_NAME] ([machine_name]\[user_name])
            Web Sites
                    Default Web Site
```

7. In the right panel, select **Add Native Module**.
8. Select **CSPms** and select **OK**.

2.2.2.2 Mapping the CSP File Extensions

Note: Do NOT use `Add Wildcard Script Mapping` utility for this file extension mapping process; it will give you an error! Instead, use the utility called `Add Module Mapping for *`.

Map the CSP file extensions to the CSP Gateway Native Modules as follows:

Extension	Native Module	Binary
*.csp	CSPms	C:\inetpub\CSPGateway\CSPms.dll
*.cls	CSPms	C:\inetpub\CSPGateway\CSPms.dll
*.zen	CSPms	C:\inetpub\CSPGateway\CSPms.dll
*.cxw	CSPms	C:\inetpub\CSPGateway\CSPms.dll

1. Open the **Internet Information Services (IIS) Manager** window.
2. In the left panel, expand the top level and expand **Web Sites**, then the **Default Web Site** section. Highlight **Default Web Site**:

```
[MACHINE_NAME] ([machine_name]\[user_name])
            Web Sites
                    Default Web Site
```

Note: This activates CSP for the whole web site. To restrict the use of CSP to specific virtual sub-directories (such as /csp/) focus control on the appropriate subdirectory (under **Default Web Site**) before creating the mappings. Repeat the process for each virtual subdirectory from which CSP content is to be served.

3. In the middle panel, double-click the **Handler Mappings** icon.
4. In the right panel, select **Add Module Mapping**.
5. In the **Add Module Mappings** dialogue, enter the following details:

   Request Path: *.csp

   Module: (select **CSPms** from the drop-down)

   Name:CSPGateway_csp
6. Select **Request Restrictions** and ensure that the box is *not* checked next to **Invoke handler only if request is mapped to**.
7. Select **OK** to return to the **Add Module Mappings** dialogue and select **OK** again.
8. Repeat the above process to add the following Module Mappings:

   Request Path: *.cls

   Module: (select CSPms from the list)

   Name:CSPGateway_cls

   Request Path: *.zen

   Module: (select CSPms from the list)

   Name:CSPGateway_zen

   Request Path: *.cxw

   Module: (select CSPms from the list)

   Name: CSPGatewayManagement

2.2.2.3 Registering Additional File Types with CSP

To configure additional file types to be processed by CSP, replicate the configuration created for the usual file extensions (.csp, .cls, .zen) for the new file extension(s).

If you need to serve other static files through the CSP Gateway or need to access the Management Portal through this web server, add mappings for file types .jpg, .gif, .png, .css, and .js.

To map requests for all files to CSP for a given path, set up the following wildcard entry for that path:

Extension	Native Module	Binary
*	CSPms	C:\inetpub\CSPGateway\CSPms.dll

2.2.2.4 Operating and Managing the Gateway

To access the CSP Gateway's systems management suite, point your browser at the following location:

http://<ip_address>/csp/bin/Systems/Module.cxw

The CSP engine is automatically invoked for requested files that contain a .csp, .cls, or .zen extension. For example:

http://<ip_address>/csp/samples/menu.csp

If you see an unauthorized user error message, refer to the security notes in the section "CSP Gateway and Security".

2.2.3 Configuring IIS to Return SOAP Fault Details

A Caché web service that encounters an error may return an `HTTP 500` error without the associated SOAP fault details. By default, IIS returns extended error information only to local clients. However, you can modify this behavior in the `<httpErrors>` element within the configuration file web.config. To do so, add the following section to instruct IIS to dispatch detailed error information to all clients.

```
<configuration>
    <system.webServer>
       <httpErrors errorMode="Detailed" />
    </system.webServer>
</configuration>
```

Use caution with this approach as sensitive information about the hosting environment may be revealed to clients. An alternative approach that avoids the security concerns of using `errorMode="Detailed"` is to instead use the `existingResponse="PassThrough"` directive.

```
<configuration>
    <system.webServer>
       <httpErrors existingResponse="PassThrough" />
    </system.webServer>
</configuration>
```

Restart IIS after making changes to the configuration.

You can make these changes manually to the IIS web.config file. Or, for a better, less error prone, approach, use the **Configuration Editor** built into the **IIS Manager**.

1. In the IIS Manager, from the **Connections** panel on the left, select the path which corresponds to the web service. For example: **Default Web Site**, then csp.
2. In the middle panel, under the section heading **Management** at the bottom, double-click on **Configuration Editor**.
3. In the **Configuration Editor** drop-down at the top labeled Section, expand system.webServer and click httpErrors.
4. Click on the value next to `existingResponse` and use the drop-down to view the options. Select `PassThrough`.
5. In the **Actions** pane on the right, click **Apply**.
6. Restart IIS after making changes to the configuration.

Further information about error handling in IIS can be found at:

http://www.iis.net/configreference/system.webserver/httperrors

2.3 Microsoft IIS 6 or Earlier

IIS is supplied with the server-oriented Windows Operating Systems (such as Windows NT Server/2000/2003). Windows XP Professional, though predominantly a client-oriented Operating System, also includes the IIS server. A web server is not supplied with Windows XP Home edition.

This book assumes that the CSP Gateway components are installed in the following directory:

C:\Inetpub\CSPGateway

If your CSP Gateway web server components are installed in a different directory, amend the directions in the following sections, as appropriate.

2.3.1 Install Locations for IIS 6

Install the CSP Gateway components and the CSP static files as follows:

1. The default location for the Native Modules:
   - CSPms.dll (Runtime module)
   - CSPmsSys.dll (Systems Management module)

   is:

   C:\Inetpub\CSPGateway

   The configuration file (CSP.INI) and Event Log (CSP.LOG) are written in this directory for non NSD-based connectivity options.
2. HyperEvents Components
   - CSPBroker.js
   - CSPxmlhttp.js(Caché version 5.1 and later)

   The default location for these files is:

   C:\cache-install-dir\csp\broker
3. Miscellaneous static resources used by the CSP Samples

   A number of static web resources (such as image files) are required by the CSP Samples. The default location for these files is:

   C:\cache-install-dir\csp\samples
4. Miscellaneous static resources used by the Management Portal (Caché v5.1 and later)

   A number of static web resources (such as image files) are required by the Management Portal. The default location for these files is:

   C:\cache-install-dir\csp\sys

2.3.2 Recommended Option: IIS and ISAPI Modules (CSPms.dll)

If you are using the ISAPI modules with the IIS web server, follow the directions in this section:

Configure the web server so that it recognizes CSP requests (files of type .csp, .cls, and .zen) and passes them to the CSP Gateway for processing.

2.3.2.1 Internet Information Services with ISAPI

If you are running any version of IIS using the ISAPI modules, follow these directions:

1. Open the **Internet Services Manager**, which in most versions of Windows is in Administrative Tools.
2. Expand the **Web Sites** folder and navigate to **Default Web Site**.
3. Right-click **Default Web Site** and select **Properties**.
4. Select the **Home Directory** tab.
5. Select **Configuration**.
6. Select the **Mappings** tab.

7. Select **Add** to display the **Add/Edit Application Extension Mapping** dialog box and add the following record:
   - Executable: \Inetpub\CSPGateway\CSPms.dll
   - Extension: `csp`
   - Verbs: Select **All Verbs**
   - **Script engine** check box: Select
   - **Check that file exists** check box: Clear
8. Repeat the above process to add the following record:
   - Executable: \Inetpub\CSPGateway\CSPms.dll
   - Extension: `cls`
   - Verbs: Select **All Verbs**
   - **Script engine** check box: Select
   - **Check that file exists** check box: Clear
9. Repeat the above process to add the following record:
   - Executable: \Inetpub\CSPGateway\CSPms.dll
   - Extension: `zen`
   - Verbs: Select **All Verbs**
   - **Script engine** check box: Select
   - **Check that file exists** check box: Clear
10. Repeat the above process to add the following record:
    - Executable: \Inetpub\CSPGateway\CSPms.dll
    - Extension: `cxw`
    - Verbs: Select **All Verbs**
    - **Script engine** check box: Select
    - **Check that file exists** check box: Clear
11. Return to **Internet Information Services** and navigate to **Default Web Site** again.
12. Right-click **Default Web Site**, point to **New** and then select **Virtual Directory**. Create a virtual directory using the following information:
    - Alias: `csp`
    - Directory: *install-dir*\csp
    - Access Permissions: Select the **Execute** check box
13. Select **Save** and restart IIS to apply your changes.

2.3.2.2 Internet Information Services v6 with ISAPI

If you are running IIS 6, follow the directions in the previous section and also the directions in this section:

This version of IIS is shipped with Windows Server 2003. To configure CSP to work with this server, register the CSP Gateway ISAPI DLLs (CSPms.dll and CSPmsSys.dll) as allowed Web service extensions.

Important: It is a common mistake to register the Gateway's modules, which are ISAPI extensions, as ISAPI filters. If you register the modules as ISAPI filters, CSP does not work.

1. Open the **Internet Services Manager**.
2. Navigate to **Web Service Extensions**. This displays a list of currently configured extensions (or applications) in the right-hand panel.
3. Right-click **Web Services Extensions** and select **Add a new Web service extension**.
4. Enter `CSP Gateway` for the **Extension** name field.
5. Select **Add**.
6. Add CSPms.dll (including the full physical path to this DLL). Repeat the process for CSPmsSys.dll (Gateway builds 999 and earlier).
7. Select the **Set extension status to Allowed** check box.
8. Select **OK**.

Note that there is an option to allow users access to all ISAPI extensions: **Allow All unknown ISAPI extensions**. Enabling this option automatically enables access to the CSP Gateway's ISAPI modules. However, to maintain security it recommended that you follow the procedure above and grant additional access only to the CSP Gateway modules.

Later, you can perform the following additional operations on registered Web Service Extensions. IIS 6 lets you turn off aspects of access to CSP.

To Prohibit Access to CSP Web Gateway Management Page

Use this procedure to disable access to the CSP Web Gateway Management page available from the Management Portal. Doing this prevents the possibility of unauthorized users gaining access the CSP Web Gateway Management page for an operational system. It is a quick and straightforward procedure for system administrators to re-enable access for a future period of time in order for configuration changes to be made to the Gateway.

1. Open the **Internet Services Manager** .
2. Navigate to **Web Service Extensions** to display a list of currently configured extensions (or applications) in the right-hand panel.
3. In the right-hand window, double-click **CSP Gateway** to display the **Web Service Extension Properties** window.
4. Select the **Required Files** tab.
5. Select CSPmsSys.dll to select this file.
6. Select Prohibit; select **Apply**; select **OK**.

You can also use this procedure to prevent end-users from gaining access to CSP resources while significant changes are being made to the Gateway configuration. In this case, the Gateway runtime module (CSPms.dll) should be marked as `Prohibited` instead of the Systems Management module (CSPmsSys.dll).

To reactivate the CSP Gateway Systems Management module, at the last step, select **Allow** instead of **Prohibit**.

2.3.2.3 Security Settings with ISAPI

For many Windows installations (particularly Windows 2000 and later), the default privileges assigned to the IIS web server are not sufficient to allow the CSP Gateway to read from and/or write to its configuration and log files (CSP.ini and CSP.log respectively).

You must, therefore, assign the web server read/write privileges to the CSP Gateway files, or grant the web server Administrator privileges. If you fail to do this, you may not be able to save your configuration changes through the CSP Web Gateway Management page.

File-access privileges can be modified through Windows Explorer. Alternatively, you can use the following two commands at the command prompt. Note that the CSP.ini and CSP.log files are in the same directory as the Gateway binaries that the web server is configured to use. With ISAPI, this is typically Inetpub.

```
cacls c:\Inetpub\CSPGateway\CSP.ini /E /G IUSR_xxx:F
cacls c:\Inetpub\CSPGateway\CSP.log /E /G IUSR_xxx:F
```

Where `IUSR_xxx` is the web server's user authority and the `xxx` component is usually the computer name (see the numbered procedure below to find the correct name).

The files that you are running the `cacls` command on must already exist. If they do not, use the `copy con` command (in a Windows Command Prompt window or DOS box) to create empty files:

```
Copy con c:\Inetpub\CSPGateway\CSP.ini
^Z
Copy con c:\Inetpub\CSPGateway\CSP.log
^Z
```

Each individual command line is terminated with carriage return. `^Z` refers to `Ctrl-Z`, which ends the `copy` command.

Example: Use the following commands to adjust the CSP Gateway configuration and log file access rights for a computer named BOSTON:

```
cacls c:\Inetpub\CSPGateway\CSP.ini /E /G IUSR_BOSTON:F
cacls c:\Inetpub\CSPGateway\CSP.log /E /G IUSR_BOSTON:F
```

You can find the specific name to use in the Internet Service Manager by navigating to the `Authentication methods` dialog as follows:

1. Open the **Internet Services Manager**.
2. Navigate to **Default Web Site**.
3. Right-click **Default Web Site** and select **Properties**
4. Select the **Directory Security** tab.
5. Select **Edit** under the **Anonymous access and authentication** control panel. This displays the **Username in the Authentication methods** dialog box.

2.3.2.4 Registering Additional File Types with CSP

To configure additional file types to be processed by CSP, replicate the configuration created for the usual file extensions (that is,.csp, .cls, .zen) for the new file extension(s).

If you need to serve other static files through the CSP Gateway or need to access the Management Portal through this web server, add mappings for file types .jpg, .gif, .png, .css, and .js.

To map requests for all files to CSP for a given path, set up the following wildcard entry for that path:

Executable:	c:\cache-install-dir\csp\bin\CSPms.dll
Extension:	.* (dot asterisk)
All Verbs:	Check
Script engine:	Check
Check that file exists:	UnCheck

If the above does not work for your operating system, do the following:

1. Open the **Internet Services Manager**.
2. Navigate to **Default Web Site**, right-click and select **Properties**.
3. Select the **Home Directory** tab and select **Configuration**.
4. Under **Mappings** tab, insert an asterisk.
5. For Executable, select CSPms.dll and clear the **Verify that file exists**.

2.3.2.5 Operating and Managing the Gateway with ISAPI

To access the CSP Web Gateway Management page, enter one of the following URLs in your browser:

```
http://localhost:<port_no>/csp/bin/Systems/Module.cxw
http://localhost:<port_no>/csp/bin/CSPmsSys.dll.
```

If you see an `Unauthorized User` error message, refer to the section on "security considerations".

The CSP engine is automatically invoked for requested files with names that end in .csp or .cls. For example:

```
http://localhost:<port_no>/csp/samples/menu.csp
```

2.4 Apache Servers

Apache is supplied by the Apache Group and can be downloaded free of charge from: http://www.apache.org.

The complete source code to Apache is available from Apache for download together with clear instructions for building the server. To build Apache under Windows, you must have the Microsoft C compiler (Visual C++) version 5.0 (or later). Instead of building the server yourself, you can instead download prebuilt kits for Windows. The prebuilt kits are, generally, a few builds behind the latest Apache source code.

This guide assumes that the CSP Gateway components are installed in the following directory:

C:\Program Files\Apache Group\Apache\CSPGateway

It is assumed that the web server is installed under:

C:\Program Files\Apache Group\Apache\

If the layout is different on your system, amend the configuration directives in the following sections, as appropriate.

First follow the directions in the section "Install Locations with Apache Servers (All Options)", then follow the directions in the section "Recommended Option: Apache API Modules (CSPa.dll) " section that follows or, if you are installing an atypical configuration, see the appendix Alternative Configurations for Microsoft Windows.

2.4.1 Install Locations with Apache Servers (All Options)

All users of the Apache server should follow the directions in this section.

Install the CSP Gateway components and the CSP static files as follows:

1. CGI and other dynamically-linked modules:

   The common files for all Apache versions are:

   - CSPcgi.exe (Runtime module)
   - nph-CSPcgi.exe (Copy of CSPcgi)

- CSPcgiSys.exe (Systems-Management module)
- nph-CSPcgiSys.exe (Copy of CSPcgiSys)

Note: There are separate binaries for each version of the Apache server as shown below.

Apache Version 2.4.x

- mod_csp24.dll (Apache built-in momod_dule as a DLL, if supplied)
- CSPa24.dll (Runtime module, if supplied)
- CSPa24Sys.dll (Gateway Systems Management module, if supplied)

Apache Version 2.2.x

- mod_csp22.dll (Apache built-in momod_dule as a DLL, if supplied)
- CSPa22.dll (Runtime module, if supplied)
- CSPa22Sys.dll (Gateway Systems Management module, if supplied)

Apache Version 2.0.x:

- mod_csp2.dll (Apache built-in module as a DLL, if supplied)
- CSPa2.dll (Runtime module, if supplied)
- CSPa2Sys.dll (Gateway Systems Management module, if supplied)

The default location for these binaries is:

C:\Program Files\Apache Group\Apache\CSPGateway\bin

The original location (*install-dir*\csp\bin) is used to hold the Gateway components required for serving the Management Portal for the specific instance of Caché.

The configuration file (CSP.INI) and Event Log (CSP.LOG) are written in this directory for non NSD-based connectivity options.

The modules with Sys appended are special modules for accessing the CSP Web Gateway Management page. The runtime modules (that is, those without Sys) have no access to the systems management forms.

2. HyperEvents Components

   - CSPBroker.js
   - CSPxmlhttp.js

   The default location for these files is:

   C:\cache-install-dir\csp\broker

3. Miscellaneous static resources used by the CSP Samples

   A number of static web resources (such as image files) are required by the CSP Samples. The default location for these files is:

   C:\cache-install-dir\csp\samples

4. Miscellaneous static resources used by the Management Portal

   A number of static web resources (such as image files) are required by the Management Portal. The default location for these files is:

C:\cache-install-dir\csp\sys

2.4.2 Recommended Option: Apache API Modules (CSPa24.dll)

This is the option that is used by the Private Web Server that serves the Management Portal.

This connectivity option is relatively new and offers the best performance as well as being the easiest to configure. Apache under Windows is entirely multithreaded and its modules persist in memory from the time Apache is started. These two essential characteristics make it possible to implement the Gateway's functionality as a set of stand-alone modules.

If you are installing an atypical configuration, see the appendix Alternative Configurations for Microsoft Windows

The modules CSPap24.dll (Runtime) and CSPapSys24.dll (Gateway systems management) are dynamically-linked modules that are designed to work the same way as the corresponding Microsoft ISAPI DLLs. These modules are named according to their Apache version:

- Apache 2.4.x: Use modules CSPa24.dll and CSPa24Sys.dll.
- Apache 2.2.x: Use modules CSPa22.dll and CSPa22Sys.dll.
- Apache 2.0.x: Use modules CSPa2.dll (Runtime) and CSPa2Sys.dll

Configure the web server so that it recognizes CSP requests (files of type .csp, .cls, and .zen) and passes them to the CSP Gateway module for processing.

The web server configuration file (httpd.conf) is in the following directory:

C:\Program Files\Apache Group\Apache\conf

1. Apache 2.4.x: Add the section below to the end of httpd.conf.

```
LoadModule csp_module_sa c:/cache-install-dir/csp/bin/CSPa24.dll
<Location "/csp/bin/Systems/">
SetHandler csp-handler-sa
</Location>
<Location "/csp/bin/RunTime/">
SetHandler csp-handler-sa
</Location>
CSPFileTypes csp cls zen cxw
Alias /csp/ c:/cache-install-dir/csp/
<Directory "c:/cache-install-dir/csp">
    AllowOverride None
    Options MultiViews FollowSymLinks ExecCGI
    Require all granted
    <FilesMatch "\.(log|ini|pid|exe)$">
    Require all denied
    </FilesMatch>
</Directory>
```

Apache 2.2.x: Add the section below to the end of httpd.conf

```
LoadModule csp_module_sa c:/cache-install-dir/csp/bin/CSPa22.dll
<Location "/csp/bin/Systems/">
SetHandler csp-handler-sa
</Location>
<Location "/csp/bin/RunTime/">
SetHandler csp-handler-sa
</Location>
CSPFileTypes csp cls zen cxw
Alias /csp/ c:/cache-install-dir/csp/
<Directory "c:/cache-install-dir/csp">
    AllowOverride None
    Options MultiViews FollowSymLinks ExecCGI
    Order allow,deny
              Allow from all
    <FilesMatch "\.(log|ini|pid|exe)$">
    Deny from all
    </FilesMatch>
</Directory>
```

Apache 2.0.x: Add the section above using CSPa2.dll to the end of httpd.conf

2. Restart Apache after making changes to httpd.conf.

2.4.2.1 Registering Additional File Types with CSP

Apache API modules always recognize the following reserved file extensions:

```
.csp .cls .zen .cxw
```

You may have other files that you want to send to CSP for processing. For example, if you need to serve other static files through the CSP Gateway or need to access the Management Portal through this web server, add mappings for file types .jpg, .gif, .png, .css, and .js.

You can configure Apache to recognize what files to pass on to CSP in any of the following ways:

- By `CSP` location directive
- By file extension—`CSPFileTypes` directive
- By MIME type

By CSPlocation directive

Use the CSP directive to request that all files within a certain location be processed by CSP. The following requests that all files and directories under the /csp path be processed by CSP.

```
<Location /csp>
   CSP On
   SetHandler csp-handler-sa
</Location>
```

For example, all the following would be sent to CSP for processing:

```
/csp/
csp/samples/menu.csp
csp/sys/
```

By file extension — CSPFileTypes directive

The `CSPFileTypes` directive works for requests for files that have extensions (such /csp/menu.csp). It does not work for requests for files that do not have file extensions (such as/csp/menu).

This parameter is processed by the Gateway's Apache modules and can be globally defined at the server definition level (in httpd.conf) or restricted within the definition for a location or directory block.

By file type: The following directive requests that files of type xxx and yyy be processed by CSP.

```
CSPFileTypes xxx yyy
```

By location: The following requests that files of type xxx and yyy be processed by CSP but only for locations under /csp (including subdirectories, such as /csp/samples and so on).

```
<Location /csp/>
   CSPFileTypes xxx yyy
</Location>
```

Using the wildcard character, the following requests that all files under path /csp (and /csp/samples and so on) be processed by CSP.

```
<Location /csp/>
   CSPFileTypes *
</Location>
```

By MIME type

In addition to recognizing the file extensions listed above, CSP can also recognize files for the following MIME types:

```
application/x-csp
```

and

```
text/csp
```

For example, to add the file extension xxx to the list of files processed by CSP, use:

```
LoadModule csp_module_sa /cache-install-dir/csp/bin/CSPa24.dll
AddType application/x-csp csp cls zen xxx
```

One of the problems with using MIME types to associate types of file with CSP is that Apache checks to ensure that the path to the resource (that is, the hosting directory) physically exists, and returns a `file not found` error if it does not. It does not, however, check to ensure that the file requested physically exists – which is appropriate for resources served by CSP since they are served by Caché and are virtual as far as the web server is concerned. The "By MIME type" approach is therefore only suitable for cases where the application's path structure can be replicated on the web server.

2.4.2.2 Operating and Managing the Gateway with Apache API

To access the CSP Gateway's CSP Web Gateway Management page, point your browser at:

```
http://localhost:<port_no>/csp/bin/Systems/Module.cxw
```

Notice the use of the cxw file extension. This extension prevents Apache attempting to load and run these DLLs through the Apache Group ISAPI interface. Also, remember that URL paths and files names are case-sensitive under Apache.

If you see an `Unauthorized User` error message, refer to the section on "security considerations".

The CSP engine is automatically invoked for requested files that contain a .csp, .cls, or .zen extension. For example:

```
http://localhost:<port_no>/csp/samples/menu.csp
```

2.5 Nginx Servers

Nginx is an Open Source product. The source code can be downloaded free of charge from:

http://nginx.org/

Some prebuilt kits are available for Windows which are, generally, a few builds behind the latest Nginx build. However, given that extensions must be compiled into the Nginx core, it is necessary to build the web server locally from the source code in order to include support for CSP.

This guide is based on CSP/Gateway web server components being installed under the following file system:

C:\cachesys\csp\

It is assumed that the web server is installed under:

C:\nginx\

If the layout is different on your system, be sure to amend the configuration directives described in the following sections, as appropriate.

2.5.1 Installation

The CSP Gateway components and the CSP static files should be installed as follows:

1. Dynamically linked Universal CSP Gateway Modules
   - CSPx.dll (Run-time module)

- CSPxSys.dll (Gateway Systems Management module)

The default location for these binaries is:

C:\cachesys\csp\bin

The configuration file (CSP.INI) and Event Log (CSP.LOG) are written in this directory.

The modules with Sys appended are special modules for accessing the CSP systems management suite. The run-time modules (that is, those without Sys) have no access to the systems management forms.

2. HyperEvents Components

   - CSPBroker.js
   - CSPxmlhttp.js

   The default location for these files is:

   C:\cachesys\csp\broker

   If these files are to be served as static components directly by the web server they should be copied to the following location:

   C:\nginx\html\csp\broker

3. Miscellaneous static resources used by the CSP Samples

   A number of static web resources (such as image files) are required by the CSP Samples. The default location for these files is:

   C:\cachesys\csp\samples

   Copy these to the following location if they are to be served directly by the web server:

   C:\nginx\html\csp\samples

4. Miscellaneous static resources used by the Caché SMP

   A number of static web resources (such as image files) are required by the SMP. The default location for these files is:

   C:\cachesys\csp\sys

   Copy these to the following location if they are to be served directly by the web server:

   C:\nginx\html\csp\sys

   See the section Static Files in *Using CSP* for more information.

2.5.2 Building the Nginx web server for CSP

Most of the CSP Gateway functionality is provided by the Universal Modules (CSPx[Sys].dll). For CSP access, Nginx can be built and configured to communicate with these Universal Modules through a small compiled-in module:

ngx_http_csp_module_sa.c

Prerequisites for building Nginx:

- Microsoft Visual Studio (preferably version 10).

  http://www.microsoft.com

- MSYS (from MinGW).

  http://www.mingw.org/wiki/MSYS

- Perl (preferably ActivePerl).

  http://www.activestate.com/activeperl

- Mercurial source control client.

  http://mercurial.selenic.com/

The build instructions given here are based on the official documentation for building Nginx under Windows:

http://nginx.org/en/docs/howto_build_on_win32.html

The Nginx documentation stipulates that the following third party add-ons are also required:

- PCRE

  http://www.pcre.org/

- OpenSSL (for SSL/TLS)

  http://www.openssl.org/

- Zlib

  http://zlib.net/

However, it is possible to create a fully functional server without these components, provided the final installation doesn't require the functionality that would otherwise be provided by them.

The default configuration script for building Nginx, including all optional modules listed above, is as follows:

```
auto/configure --with-cc=cl --builddir=objs --prefix= \
--conf-path=conf/nginx.conf --pid-path=logs/nginx.pid \
--http-log-path=logs/access.log --error-log-path=logs/error.log \
--sbin-path=nginx.exe \
--http-client-body-temp-path=temp/client_body_temp \
--http-proxy-temp-path=temp/proxy_temp \
--http-fastcgi-temp-path=temp/fastcgi_temp \
--with-cc-opt=-DFD_SETSIZE=1024 --with-pcre=objs/lib/pcre-8.32 \
--with-zlib=objs/lib/zlib-1.2.7 \
--with-openssl=objs/lib/openssl-1.0.1e \
--with-select_module --with-http_ssl_module --with-ipv6
```

The build process can be modified to exclude optional modules:

- OpenSSL - Remove SSL/TLS capability:

  Remove directive: `--with-http_ssl_module`

- Zlib - Remove GZIP capability:

  Add directive: `--without-http_gzip_module`

- PCRE - Remove HTTP rewrite capability:

  Add directive: `--without-http_rewrite_module`

2.5.2.1 Procedure for building Nginx for CSP

1. Working in a MSYS shell, create the working directory structure suggested in the Nginx documentation:

   /opt/

2. Working in /opt, check-out the Nginx source code using the following command:

   ```
   hg clone http://hg.nginx.org/nginx
   ```

   This places the Nginx source code under: /opt/nginx/

3. Create a directory for the CSP extension:

   /opt/nginx/objs/lib/csp/

4. Copy the module source code (ngx_http_csp_module_sa.c and cspapi.h) to the directory created in the previous step.

5. In the same directory, create a configuration file called config. This file should contain the following lines:

```
ngx_addon_name=ngx_http_csp_module_sa
HTTP_MODULES="$HTTP_MODULES ngx_http_csp_module_sa"
NGX_ADDON_SRCS="$NGX_ADDON_SRCS $ngx_addon_dir/ngx_http_csp_module_sa.c"
```

6. Working in /opt/nginx/, configure the Nginx build environment:

```
auto/configure --with-cc=cl --builddir=objs --prefix= \
--conf-path=conf/nginx.conf --pid-path=logs/nginx.pid \
--http-log-path=logs/access.log --error-log-path=logs/error.log \
--sbin-path=nginx.exe \
--http-client-body-temp-path=temp/client_body_temp \
--http-proxy-temp-path=temp/proxy_temp \
--http-fastcgi-temp-path=temp/fastcgi_temp \
--with-cc-opt=-DFD_SETSIZE=1024 --without-http_rewrite_module \
--without-http_gzip_module \
--with-select_module --with-ipv6 \
--add-module=objs/lib/csp
```

   Note the final line containing the instructions to include the CSP module.

7. Compile Nginx:

```
nmake -f objs/Makefile
```

   If successful, you should find the server (nginx.exe) in: /opt/nginx/objs/

8. Install Nginx:

   The easiest way to do this is to first download and install a pre-built version of Nginx for Windows to obtain the directory structure (usually under C:\nginx\) then replace the nginx.exe file in that installation with the one created locally.

   The typical directory structure for an operational Nginx installation is as follows:

```
Directory of C:\nginx

03/07/2013  09:09    <DIR>          .
03/07/2013  09:09    <DIR>          ..
26/06/2013  10:14    <DIR>          conf
26/06/2013  10:14    <DIR>          contrib
10/05/2014  12:53    <DIR>          csp
26/06/2013  10:14    <DIR>          docs
26/06/2013  10:14    <DIR>          html
10/05/2014  15:57    <DIR>          logs
04/07/2013  15:52           715,264 nginx.exe
26/06/2013  10:17    <DIR>          scgi_temp
26/06/2013  10:17    <DIR>          temp
26/06/2013  10:17    <DIR>          uwsgi_temp
```

   Replace the copy of *nginx.exe* in this directory with the version created by the build procedure.

2.5.3 Using the Universal CSP Gateway Modules (CSPx*.dll)

The Universal Modules CSPx.dll (Run-time) and CSPxSys.dll (Gateway systems management) are dynamically linked modules that are designed to be loaded by a CSP-enabled Nginx installation.

The web server should be configured such that it recognizes CSP requests (files of type .csp, .cls, and .zen) and passes them to the CSP Gateway module for processing

The web server configuration file (nginx.conf) is found in the following directory:

C:\nginx\conf

The following configuration directives are provided for the CSP extension:

- CSPModulePath

  http section: Path to the Universal Gateway Modules

- CSP

  server section: Enable CSP for an entire path

- CSPFileTypes

  server section: Enable CSP for specific file types

For Windows, the thread stack size must be increased to 2MB. To do this, add the following directive to the top of the Nginx configuration file (before the **http** section):

```
thread_stack_size 2000000;
```

Place the following directive within the **http** configuration block:

```
CSPModulePath C:/cachesys/csp/bin/;
```

This directive enables Nginx to find the Universal Gateway Modules (CSPx[Sys].dll) and the associated configuration (CSP.ini).

Nginx can now be configured to pass all requests for a certain path to CSP or just certain file types.

2.5.3.1 Enabling CSP for a Particular Path

Place the following section within the appropriate **server** configuration block:

```
location /csp {
CSP On;
```

2.5.3.2 Registering Specific File Types with CSP

Place the following section within the appropriate **server** configuration block:

```
location /csp {
CSPFileTypes  csp cls zen cxw;
```

2.5.3.3 Starting and Stopping Nginx

First, ensure that Nginx has read/write permissions to the location holding the Universal Gateway Modules (C:/cachesys/csp/bin/). This is the location where the Gateway configuration and Event Log files will be maintained (CSP.ini and CSP.log).

To start Nginx:

```
C:\nginx\nginx
```

To stop Nginx:

```
C:\nginx\nginx –s stop
```

2.5.3.4 Operating and Managing the Gateway

To access the CSP Gateway's systems management suite, point your browser:

```
http://<ip_address>/csp/bin/Systems/Module.cxw
```

Notice the use of the cxw file extension.

The CSP engine is automatically invoked for requested files containing the .csp, .cls, or .zen extensions. For example:

```
http://<ip_address>/csp/samples/menu.csp
```

If you see an `Unauthorized User` error message, refer to the section on security considerations in this book.

2.5.4 Building Nginx to Work with the CSP NSD Component

Nginx can be built to work with the CSP Gateway's *Network Service Daemon* component (CSPnsd[Sv].exe), as opposed to the *Universal Modules* (CSPx[Sys].dll). To do this, modify the build and configuration procedure as follows:

- Build Nginx with source code module ngx_http_csp_module.c instead of ngx_http_csp_module_sa.c
- The configuration file for CSP (/opt/nginx/objs/lib/csp/config) should read as follows:

```
ngx_addon_name=ngx_http_csp_module
HTTP_MODULES="$HTTP_MODULES ngx_http_csp_module"
NGX_ADDON_SRCS="$NGX_ADDON_SRCS $ngx_addon_dir/ngx_http_csp_module.c"
```

- Remove the **CSPModulePath** directive from the Nginx configuration.
- Start the NSD component.

The usual reason for preferring to use the NSD component is to obtain full and reliable support for CSP state-aware mode (Preserve mode). However, in order to get the best results with Nginx, it is recommended that applications should be entirely state-less.

3

Web Servers for UNIX®, Linux, and Mac OS X

Web Servers from Apache and Sun are described in this section.

Several connectivity options are available for Apache. The Apache Group provides support for extensions implemented as dynamically linked modules (DSOs). Extensions, written as Apache modules, can be built directly into the Apache core. This is the recommended option. Other, atypical, configuration options are in the appendix Alternative Configurations for UNIX®, Linux, and Mac OS X

Sun web servers can be extended using a high-performance API. You can use the Netscape Application Programming Interface (NSAPI) to extend the web server through modules implemented as UNIX® Shared Objects (or Shared Libraries).

3.1 Apache Servers

Apache is supplied by the Apache Group and can be downloaded free of charge from http://www.apache.org.

Pre-built kits are available for some UNIX® systems which are, generally, a few builds behind the latest version. The complete source code to Apache is available for download together with clear instructions for building the Apache server. The freely available GNU C compiler (gcc) can be obtained for this purpose, though the Apache build procedure attempts to use the indigenous C compiler.

Many systems are shipped with Apache pre-installed, configured and ready to go. Most distributions of Linux include Apache. IBM distributes Apache with their UNIX® implementation: AIX®.

Note: The build of apache that IBM supplies with AIX is not compatible with the recommended Apache API modules. Follow the instructions in the appendix Building Apache for IBM AIX to build a compatible version of Apache on AIX. The configurations using NSD do not require this step. For more information, see the appendix Alternative Configurations for UNIX®, Linux, and Mac OS X.

This guide refers to the directory that CSP Gateway components are installed in as:

```
/opt/cspgateway/csp/
```

This guide refers to the directory that Apache is installed in as the following; your Apache install directory may be different.

```
/usr/apache/
```

If the layout is different on your system, be sure to amend the configuration directives described in the following sections, as appropriate.

This section describes the recommended option for installing the CSP Gateway.

1. Everyone should follow the directions in the section "Install Locations with Apache Servers on UNIX (All Options)".
2. Then follow the directions in the section "Recommended Option: Apache API Module without NSD (CSPa.so)". Or, if you are using an atypical option, see the appendix Alternative Configurations for UNIX, Linux, and Mac OS.

3.1.1 Install Locations Apache UNIX®, Linux, Mac OS (Recommended Option)

This section describes directory locations for CSP Gateway files and CSP static files.

1. Dynamically-linked modules:
   - CSPa24.so (Apache Version 2.4.x)
   - CSPa22.so (Apache Version 2.2.x)
   - CSPa20.so (Apache Version 2.0.x)

   In order to avoid disrupting existing Gateway installations on upgrading Caché, the installation procedures place these modules in the following common location. This location is not related to a particular Caché instance.

   ```
   /opt/cspgateway/bin
   ```

   The original location (/cache-install-dir/csp/bin) is used to hold the Gateway components required for serving the Management Portal for the specific instance of Caché.

   The modules with Sys appended access the CSP Web Gateway Management page. The runtime modules (that is, those without Sys) have no access to the CSP Web Gateway Management pages.
2. HyperEvents Components
   - CSPBroker.js
   - CSPxmlhttp.js

   The default location for these files is:

   ```
   /cache-install-dir/csp/broker
   ```
3. Miscellaneous static resources used by the CSP Samples

   A number of static web resources (such as image files) are required by the CSP Samples. The default location for these files is:

   ```
   /cache-install-dir/csp/samples
   ```
4. Miscellaneous static resources used by the Management Portal.

   A number of static web resources (such as image files) are required by the Management Portal. The default location for these files is:

   ```
   /cache-install-dir/csp/sys
   ```

Requirements for using Apache API Modules (Recommended Option and Alternative Option 1)

Before following instructions for either Recommended Option and Alternative Option 1, check that your build of Apache includes the built-in module for managing shared objects (mod_so). To perform this check, run the following command which lists the modules currently available within Apache:

```
httpd -l
```

The shared object module (mod_so) should appear in the list of modules displayed. The following shows a typical module listing with mod_so included:

```
Compiled in modules:
  core.c
  mod_access.c
  mod_auth.c
  mod_include.c
  mod_log_config.c
  mod_env.c
  mod_setenvif.c
  prefork.c
  http_core.c
  mod_mime.c
  mod_status.c
  mod_autoindex.c
  mod_asis.c
  mod_cgi.c
  mod_negotiation.c
  mod_dir.c
  mod_imap.c
  mod_actions.c
  mod_userdir.c
  mod_alias.c
  mod_so.c
```

If mod_so is not included in the list for your Apache installation, refer to your Apache documentation and follow the procedure for rebuilding Apache to include this module.

3.1.2 Recommended Option: Apache API Module without NSD (CSPa24.so)

This option is used in the configuration of the Private Web Server in the Management Portal.

This connectivity option offers the best performance as well as being the easiest to configure.

Before using this option you should bear in mind that Apache v2.x..x is partially multithreaded, implemented as a hybrid multi-process and multithreaded server. In practice, this means that there is one instance of the CSP Gateway per Apache child process. This is not a problem in itself but this architecture makes it difficult to control the number of connections to Caché (and Caché processes) used by the Gateway since each instance of the Gateway independently manages its own pool of Caché connections.

State-aware connectivity (preserve mode 1) should not be used with these modules.

The modules CSPa24.so (Runtime) and CSPa24Sys.so (Gateway systems management) are dynamically linked modules (DSOs).

Configure the web server to recognize CSP requests (files of type .csp, .cls, and .zen) and pass them to the CSP Gateway module for processing.

- Apache 2.4.x: Use modules CSPa24.so and CSPa24Sys.so.
- Apache 2.2.x: Use modules CSPa22.so and CSPa22Sys.so.
- Apache 2.0.x: Use modules CSPa2.so and CSPa2Sys.so

The web server configuration file (httpd.conf) is in the following directory:

```
/usr/apache/conf
```

For Red Hat Linux, the runtime version of httpd.conf is found in:

```
/etc/httpd/conf
```

For Suse, the runtime version of httpd.conf is found in:

```
/etc/apache2/conf
```

1. Apache 2.4.x: Add the section below to the end of httpd.conf.

```
LoadModule csp_module_sa /opt/cspgateway/bin/CSPa24.so
CSPModulePath /opt/cspgateway/bin/
<Location "/csp/bin/Systems/">
SetHandler cspsys-handler-sa
</Location>
<Location "/csp/bin/RunTime/">
SetHandler csp-handler-sa
</Location>
CSPFileTypes csp cls zen cxw
Alias /csp/ /opt/cspgateway/csp/
<Directory "/opt/cspgateway/csp">
         AllowOverride None
         Options MultiViews FollowSymLinks ExecCGI
         Require all granted
         <FilesMatch "\.(log|ini|pid|exe)$">
         Require all denied
         </FilesMatch>
</Directory>
```

   Apache 2.2.x: Add the section below to the end of httpd.conf.

```
LoadModule csp_module_sa /opt/cspgateway/bin/CSPa22.so
CSPModulePath /opt/cspgateway/bin/
<Location "/csp/bin/Systems/">
SetHandler cspsys-handler-sa
</Location>
<Location "/csp/bin/RunTime/">
SetHandler csp-handler-sa
</Location>
CSPFileTypes csp cls zen cxw
Alias /csp/ /opt/cspgateway/csp/
<Directory "/opt/cspgateway/csp">
         AllowOverride None
         Options MultiViews FollowSymLinks ExecCGI
         Order allow,deny
         Allow from all
         <FilesMatch "\.(log|ini|pid|exe)$">
         Deny from all
         </FilesMatch>
</Directory>
```

   Apache 2.0.x: Add the section above using CSPa2.dll to the end of httpd.conf

2. Restart Apache after making changes to httpd.conf.

3.1.2.1 Registering Additional File Types with CSP

Apache API modules always recognize the following reserved file extensions:

```
.csp .cls .zen .cxw
```

You may have other files that you want to send to CSP for processing. For example, if you need to serve other static files through the CSP Gateway or need to access the Management Portal through this web server, add mappings for file types .jpg, .gif, .png, .css, and .js.

You can configure Apache to recognize what files to pass on to CSP in any of the following ways:

- By CSP location directive
- By file extension—`CSPFileTypes` directive
- By MIME type

By CSPlocation directive

Use the CSP directive to request that all files within a certain location be processed by CSP. The following requests that all files and directories under the /csp path be processed by CSP.

```
<Location /csp>
   CSP On
   SetHandler csp-handler-sa
</Location>
```

For example, all the following would be sent to CSP for processing:

```
/csp/
csp/samples/menu.csp
csp/sys/
```

By file extension — CSPFileTypes directive

The `CSPFileTypes` directive works for requests for files that have extensions (such /csp/menu.csp). It does not work for requests for files that do not have file extensions (such as/csp/menu).

This parameter is processed by the Gateway's Apache modules and can be globally defined at the server definition level (in httpd.conf) or restricted within the definition for a location or directory block.

By file type: The following directive requests that files of type xxx and yyy be processed by CSP.

```
CSPFileTypes xxx yyy
```

By location: The following requests that files of type xxx and yyy be processed by CSP but only for locations under /csp (including subdirectories, such as /csp/samples and so on).

```
<Location /csp/>
   CSPFileTypes xxx yyy
</Location>
```

Using the wildcard character, the following requests that all files under path /csp (and /csp/samples and so on) be processed by CSP.

```
<Location /csp/>
   CSPFileTypes *
</Location>
```

By MIME type

In addition to recognizing the file extensions listed above, CSP can also recognize files for the following MIME types:

```
application/x-csp
```

and

```
text/csp
```

For example, to add the file extension xxx to the list of files processed by CSP, use:

```
LoadModule csp_module_sa /opt/cspgateway/bin/CSPa22.so
AddType application/x-csp csp cls zen xxx
```

One of the problems with using MIME types to associate types of file with CSP is that Apache checks to ensure that the path to the resource (that is, the hosting directory) physically exists, and returns a `file not found` error if it does not. It does not, however, check to ensure that the file requested physically exists – which is appropriate for resources served by CSP since they are served by Caché and are virtual as far as the web server is concerned. The "By MIME type" approach is therefore only suitable for cases where the application's path structure can be replicated on the web server.

3.1.2.2 Operating and Managing the Gateway with Apache API

To access the CSP Web Gateway Management page, point your browser to:

```
http://localhost:<port_no>/csp/bin/Systems/Module.cxw
```

Notice the use of the cxw file extension. This extension prevents Apache attempting to load and run these DLLs through the Apache Group ISAPI interface. Also, remember that URL paths and files names are case-sensitive under Apache.

If you see an `Unauthorized User` error message, refer to the section on security considerations.

The CSP engine is automatically invoked for requested files that contain a .csp, .cls, or .zen extension. For example:

```
http://localhost:<port_no>/csp/samples/menu.csp
```

If you see an `Unauthorized User` error message, refer to the section on security considerations.

3.2 Sun Web Servers

This section covers the configuration and operational procedures for running CSP through the Sun Web Server. This guide is based on the CSP web server components being installed in the following file system:

```
/cache-install-dir/csp/
```

It is assumed that the web server is installed under:

```
/usr/SUNWwbsvr/   (or /opt/SUNWwbsvr/)
```

Individual instances of the Sun server are installed under directories of the form:

```
/usr/SUNWwbsvr/https-<server_name>/
```

or

```
/usr/SUNWwbsvr/httpd-<server_name>/
```

Where *server_name* is the logical name assigned to the hosting computer.

If the layout is different on your system, be sure to amend the configuration directives described in the following sections, as appropriate.

The documentation root directory for these servers is usually:

```
/usr/SUNWwbsvr/docs/
```

3.2.1 Install Locations for Sun Web Servers (Recommended Option)

Install the CSP Gateway components and the CSP static files as follows:

1. NSAPI and CGI Modules
   - CSPn3.so (Runtime module)
   - CSPn3Sys.so (Systems-management module)
   - CSPcn3.so (NSAPI client to the NSD – if supplied)
   - CSPcgi.exe (Runtime module)
   - nph-CSPcgi.exe (Copy of CSPcgi)
   - CSPcgiSys.exe (Systems-Management module)
   - nph-CSPcgiSys.exe (Copy of CSPcgiSys)

Note that not all of the modules listed above are required for all connectivity options. Refer to the sections describing each option to see which are actually required.

The default location for these modules is:

```
/opt/cspgateway/bin
```

The configuration file (CSP.INI) and Event Log (CSP.LOG) are written in this directory for non-NSD based connectivity options.

2. HyperEvents Components
   - CSPBroker.js
   - CSPxmlhttp.js

   The default location for these files is:

   ```
   /cache-install-dir/csp/broker
   ```

3. Miscellaneous static resources used by the CSP Samples

   A number of static web resources (such as image files) are required by the CSP Samples. The default location for these files is:

   ```
   /cache-install-dir/csp/samples
   ```

4. Miscellaneous static resources used by the Management Portal.

   A number of static web resources (such as image files) are required by the Management Portal. The default location for these files is:

   ```
   /cache-install-dir/csp/sys
   ```

3.2.2 Recommended Option: NSAPI Modules (CSPn3.so)

Configure the web server to recognize CSP requests (files of type .csp, .cls, and .zen) and pass them to the CSP gateway for processing.

The web server configuration files (magnus.conf and obj.conf) are in the following directory:

```
/usr/SUNWwbsvr/https-<server_name>/config/
```

You need to add directives to load the NSAPI modules and instructions for recognizing and processing CSP files to the web server configuration.

3.2.2.1 Directives

Follow the direction in each of the directive subsections below.

Directives for Loading NSAPI Modules

The `Init` directive instructs the web server to load NSAPI modules. These directives should be added to the core magnus.conf file. These core configuration directives are always present, an example of which is as follows:

```
Init fn=load-types mime-types=mime.types
```

Locate the block of core `Init` directives and add the following section before this block:

```
Init fn=load-modules shlib=/opt/cspgateway/bin/CSPn3.so \
     funcs=csp_term,csp_init,csp_req
Init fn=csp_init shlib="/opt/cspgateway/bin/CSPn3.so"
```

Note that `Init` directives are made up of a single line. Due to the limitations of space, the lines shown above are wrapped before the function declarations (funcs).

Directives for Locating Static Components

CSP includes a number of static files that are served by the web server. For example, the Java/JavaScript files used to implement hyperevents and the image used in the CSP samples. These files are detailed in sections 2 and 3 of the installation section respectively.

The web server needs to know the location of these files relative to the virtual CSP documentation root directory.

Find the `default` directives section of obj.conf:

```
<Object name="default">
```

Add the following line in the `default` section – that is, after the line shown above.

```
NameTrans fn="pfx2dir" from="/csp/samples" dir="/cache-install-dir/csp/samples"
NameTrans fn="pfx2dir" from="/csp/broker" dir="/cache-install-dir/csp/broker"
```

Directives for Recognizing and Processing CSP Requests

Add the following section to the end of obj.conf:

```
<Object ppath="*/*.csp">
Service method=(GET|HEAD|POST) fn=csp_req
</Object>
<Object ppath="*/*.cls">
Service method=(GET|HEAD|POST) fn=csp_req
</Object>
<Object ppath="*/*.zen">
Service method=(GET|HEAD|POST) fn=csp_req
</Object>
<Object ppath="*/CSPn3Sys.so">
Service method=(GET|HEAD|POST) fn=csp_req
</Object>
<Object ppath="*/CSPn3.so">
Service method=(GET|HEAD|POST) fn=csp_req
</Object>
<Object ppath="*/Systems/Module.cxw">
Service method=(GET|HEAD|POST) fn=csp_req
</Object>
<Object ppath="*/RunTime/Module.cxw">
Service method=(GET|HEAD|POST) fn=csp_req
 </Object>
```

3.2.2.2 Registering additional file types with CSP

Configuring additional file types to be processed by CSP is simply a question of replicating, for the new file extension(s), the configuration block created (at the end of obj.conf) for the usual file extensions (csp,cls, zen).

If you need to serve other static files through the CSP Gateway or need to access the Management Portal through this web server, add mappings for file types .jpg, .gif, .png, .css, and .js.

To map requests for all files to CSP for a given path, set up a wildcard entry for that path. For example:

```
<Object ppath="/csp/*.*">
Service method=(GET|HEAD|POST) fn=csp_req
</Object>
```

It is not necessary to add any further configuration block to this section. In addition to this modification make sure that any configuration directives previously used to alias your paths to physical locations on the web server are removed. In other words, the following lines (or similar) should not be added to the obj.conf file:

```
NameTrans fn="pfx2dir" from="/csp/samples" dir="/cache-install-dir/csp/samples"
NameTrans fn="pfx2dir" from="/csp/broker" dir="/cache-install-dir/csp/broker"
```

3.2.2.3 Operating and Managing the Gateway with Sun NSAPI

The web server must be restarted after making changes to its configuration files (magnus.conf and obj.conf).

To access the CSP Web Gateway Management page, point your browser at one of the following locations:

```
http://localhost:<port_no>/csp/bin/Systems/Module.cxw
http://localhost:<port_no>/csp/bin/CSPn3Sys.so
```

If you see an `Unauthorized User` error message, refer to the section on security considerations.

The CSP engine is automatically invoked for requested files that contain a .csp, .cls, or .zen extension. For example:

```
http://localhost:<port_no>/csp/samples/menu.csp
```

3.3 Nginx Web Servers

Nginx is an Open Source product and the source code can be downloaded free of charge from:

http://nginx.org/

Some prebuilt kits are available for Linux which are, generally, a few builds behind the latest Nginx build. However, given that extensions must be compiled into the Nginx core, it is necessary to build the web server locally from the source code in order to include support for CSP.

This guide assumes that CSP/Gateway web server components are installed under the following directory:

/opt/cspgateway/bin/

It assumes that Caché, if installed locally, is under:

/opt/cachesys/

It assumes that the web server is installed under:

/opt/nginx/

If the layout is different on your system, be sure to amend the various configuration directives described in the following sections, as appropriate.

3.3.1 Installation

The CSP Gateway components and the CSP static files should be installed as follows:

1. Dynamically linked Universal CSP Gateway Modules
   - CSPx.so (Run-time module)
   - CSPxSys.so (Gateway Systems Management module)

   The default location for these binaries is:

   /opt/cspgateway/bin/

   The configuration file (CSP.INI) and Event Log (CSP.LOG) are written in this directory.

   The modules with Sys appended are special modules for accessing the CSP systems management suite. The run-time modules (that is, those without Sys) have no access to the systems management forms.
2. HyperEvents Components

   CSPBroker.js

   CSPxmlhttp.js

   The default location for these files is:

/opt/cachesys/csp/broker

If these files are to be served as static components directly by the web server they should be copied to the following location:

/opt/nginx/html/csp/broker

3. Miscellaneous static resources used by the CSP Samples

   A number of static web resources (such as image files) are required by the CSP Samples. The default location for these files is:

   /opt/cachesys/csp/samples

   Copy these to the following location if they are to be served directly by the web server:

   /opt/nginx/html/csp/samples

4. Miscellaneous static resources used by the Caché SMP

   A number of static web resources (such as image files) are required by the SMP. The default location for these files is:

   /opt/cachesys/csp/sys

   Copy these to the following location if they are to be served directly by the web server:

   /opt/nginx/html/csp/sys

   See the section Static Files in *Using CSP* for more information.

3.3.2 Building the Nginx Web Server for CSP

Most of the CSP Gateway functionality is provided by the Universal Modules (CSPx[Sys].so). For CSP access, Nginx can be built and configured to communicate with these Universal Modules through a small compiled-in module:

ngx_http_csp_module_sa.c

The build instructions given here are based on the official documentation for building Nginx under UNIX systems:

http://nginx.org/en/docs/configure.html

The Nginx documentation stipulates that the following third party add-ons are also required:

- PCRE

  http://www.pcre.org/

- OpenSSL (for SSL/TLS)

  http://www.openssl.org/

- Zlib

  http://zlib.net/

However, it is possible to create a fully functional server without these components, provided the final installation doesn't require the functionality that would otherwise be provided by them.

A typical configuration script for building Nginx, including all optional modules listed above, is as follows:

```
./configure --prefix=/opt/nginx --with-http_ssl_module
```

This results in a default Nginx build installed under: /opt/nginx

The build process can be modified to exclude optional modules:

- OpenSSL - Remove SSL/TLS capability:

  Remove directive: `--with-http_ssl_module`

- Zlib - Remove GZIP capability:

  Add directive: `--without-http_gzip_module`

- PCRE - Remove HTTP rewrite capability:

  Add directive: `--without-http_rewrite_module`

3.3.2.1 Procedure for building Nginx for CSP

1. Unpack the source distribution under a location of your choice. For example:

   /opt/

   After unpacking, if you specify /opt/, the source code distribution is under:

   /opt/nginx-n.n.n/

2. Create a directory for the CSP extension:

   /opt/nginx-n.n.n/csp/

3. Copy the module source code (ngx_http_csp_module_sa.c and cspapi.h) to the directory created above.

4. In that same directory, create a configuration file called config. This file should contain the following lines:

   ```
   ngx_addon_name=ngx_http_csp_module_sa
   HTTP_MODULES="$HTTP_MODULES ngx_http_csp_module_sa"
   NGX_ADDON_SRCS="$NGX_ADDON_SRCS $ngx_addon_dir/ngx_http_csp_module_sa.c"
   CORE_LIBS="$CORE_LIBS -ldl"
   ```

5. Working in /opt/nginx-n.n.n/, configure the Nginx build environment:

   ```
   ./configure --prefix=/opt/nginx \
   --with-http_ssl_module \
   --add-module=/opt/nginx-n.n.n/csp
   ```

   Alternatively, without the optional functionality provided by OpenSSL, ZLIB and PCRE:

   ```
    ./configure --prefix=/opt/nginx \
   --without-http_rewrite_module \
   --without-http_gzip_module \
   --add-module=/opt/nginx-n.n.n/csp
   ```

   Note the final line containing the instructions to include the CSP module.

6. Compile Nginx:

   ```
   make
   ```

7. Install Nginx:

   ```
   make install
   ```

   If successful, you should find the complete server installation under:

   /opt/nginx/

3.3.3 Using the Universal CSP Gateway Modules (CSPx*.so)

The Universal Modules CSPx.so (Run-time) and CSPxSys.so (Gateway systems management) are dynamically linked modules that are designed to be loaded by a CSP-enabled Nginx installation.

The web server should be configured such that it recognizes CSP requests (files of type .csp, .cls, and .zen) and passes them to the CSP Gateway module for processing

The web server configuration file (nginx.conf) is found in the following directory:

/opt/nginx/conf

The following configuration directives are provided for the CSP extension:

- CSPModulePath

  http section: Path to the Universal Gateway Modules

- CSP

  server section: Enable CSP for an entire path

- CSPFileTypes

  server section: Enable CSP for specific file types

Place the following directive within the **http** configuration block:

```
CSPModulePath /opt/cspgateway/bin/;
```

This directive enables Nginx to find the Universal Gateway Modules (CSPx[Sys].so) and the associated configuration (CSP.ini).

Nginx can now be configured to pass all requests for a certain path to CSP or just certain file types.

3.3.3.1 Using the Universal CSP Gateway Modules (CSPx*.dll)

The Universal Modules CSPx.dll (Run-time) and CSPxSys.dll (Gateway systems management) are dynamically linked modules that are designed to be loaded by a CSP-enabled Nginx installation.

The web server should be configured such that it recognizes CSP requests (files of type .csp, .cls, and .zen) and passes them to the CSP Gateway module for processing

The web server configuration file (nginx.conf) is found in the following directory:

C:\nginx\conf

The following configuration directives are provided for the CSP extension:

- CSPModulePath

  http section: Path to the Universal Gateway Modules

- CSP

  server section: Enable CSP for an entire path

- CSPFileTypes

  server section: Enable CSP for specific file types

For Windows, the thread stack size must be increased to 2MB. To do this, add the following directive to the top of the Nginx configuration file (before the **http** section):

```
thread_stack_size 2000000;
```

Place the following directive within the **http** configuration block:

```
CSPModulePath C:/cachesys/csp/bin/;
```

This directive enables Nginx to find the Universal Gateway Modules (CSPx[Sys].dll) and the associated configuration (CSP.ini).

Nginx can now be configured to pass all requests for a certain path to CSP or just certain file types.

Enabling CSP for a Particular Path

Place the following section within the appropriate **server** configuration block:

```
location /csp {
CSP On;
```

Registering Specific File Types with CSP

Place the following section within the appropriate **server** configuration block:

```
location /csp {
CSPFileTypes  csp cls zen cxw;
```

Starting and Stopping Nginx

First, ensure that Nginx has read/write permissions to the location holding the Universal Gateway Modules (/opt/cspgateway/bin/). This is the location where the Gateway configuration and Event Log files are maintained (CSP.ini and CSP.log).

To start Nginx:

```
/opt/nginx/sbin/nginx
```

To stop Nginx:

```
/opt/nginx/sbin/nginx –s stop
```

Operating and Managing the Gateway

To access the CSP Gateway's systems management suite, point your browser:

```
http://<ip_address>/csp/bin/Systems/Module.cxw
```

Notice the use of the cxw file extension.

The CSP engine is automatically invoked for requested files containing the .csp, .cls, or .zen extensions. For example:

```
http://<ip_address>/csp/samples/menu.csp
```

If you see an `Unauthorized User` error message, refer to the section on security considerations in this book.

3.3.4 Building Nginx to Work with the CSP NSD Component

Nginx can be built to work with the CSP Gateway's *Network Service Daemon* component (CSPnsd[Sv].exe), as opposed to the *Universal Modules* (CSPx[Sys].dll). To do this, modify the build and configuration procedure as follows:

- Build Nginx with source code module ngx_http_csp_module.c instead of ngx_http_csp_module_sa.c
- The configuration file for CSP (/opt/nginx/objs/lib/csp/config) should read as follows:

  ```
  ngx_addon_name=ngx_http_csp_module
  HTTP_MODULES="$HTTP_MODULES ngx_http_csp_module"
  NGX_ADDON_SRCS="$NGX_ADDON_SRCS $ngx_addon_dir/ngx_http_csp_module.c"
  ```

- Remove the **CSPModulePath** directive from the Nginx configuration.
- Start the NSD component.

The usual reason for preferring to use the NSD component is to obtain full and reliable support for CSP state-aware mode (Preserve mode). However, in order to get the best results with Nginx, it is recommended that applications should be entirely state-less.

4

CSP Gateway Operation and Configuration

This topic describes how to configure the CSP Gateway and exploit its functionality in CSP applications. It contains the following sections:

- CSP Web Gateway Management Page
- CSP Gateway and Security
- CGI Environment Variables
- HTTP Response Headers
- Making a CSP Page the Home Page for the Web Server
- Compressing the Response to Requests for CSP Forms (GZIP/ZLIB)
- CSP Page Output Caching
- CSP with Microsoft Active Server Pages (ASP) and VBScript
- Implementing HTTP authentication for CSP applications
- Mirrored Configurations, Failover and Load Balancing
- Process Affinity and State-Aware Mode (Preserve Mode 1)
- Using WebSockets (RFC 6455)
- Option for Automated Deployment Sites (Such As Cloud)

4.1 CSP Web Gateway Management Page

The CSP Web Gateway Management page allows you configure and manage the CSP Gateway, including monitoring its operational status. The following table shows the options available on the CSP Web Gateway Management page menu.

Menu Item	Action
System Status	Displays the status of active CSP server connections. Also allows you to clear the CSP Gateway cache.
Close Connections	Closes all connections on all or specified Caché servers.
Test Server Connection	Tests the connection to a Caché server by opening a stateless session. There is also an option to display the Caché-side Event Log.

Menu Item	Action
View Event Log	Allows you to view information in the CSP Gateway Event Log, as well as clear its contents. The Event Log is maintained on the Web Server host.
View HTTP Trace	Provides an interactive view of the HTTP requests and responses processed by the CSP Gateway.
Default Parameters	Allows you to configure the CSP Gateway on a specific web server. Also, it allows you to customize CSP responses to errors and other conditions.
Server Access	Configures CSP Gateway access to a specific Caché server.
Application Access	Configures the access to an application according to the application path. Path, in this context, refers to the path contained within the application URLs.
About CSP Gateway	Shows the CSP Gateway version, the hosting web server environment, including the path to the Gateway configuration.

The *CSP Web Gateway Documentation* (online help on configuring the CSP Gateway) is available in this section as well as through the **Help** button on the CSP Web Gateway Management Page in the Management Portal **[System] > [System Administration] > [Configuration] > [CSP Gateway Management].** If the page asks you to log in, enter your username and password (see below). Then select **Help**.

By default, this takes you to the private web server. To see the CSP Web Gateway Management page for your production web server, substitute `localhost` for `localhost:57772` in the URL, as `http://localhost:<port_no>/csp/bin/Systems/Module.cxw`.

For more information on default Caché web server port numbers, see the WebServerPort entry of the *Caché Advanced Configuration Settings Reference*.

The first time you try to access the CSP Web Gateway Management Page, you are asked for a username and password. Look for the username in the C:\Intersystems\Cache\CSP\bin\CSP.ini file. The password is the one that you entered during the Caché installation. If you forget the password, see the section "Security".

4.1.1 Localization of the CSP Web Gateway Management Page

Localization of the CSP Web Gateway Management pages is based on the contents of the CSPres.xml installed (if any). If no localization file is present then the CSP Web Gateway Management pages default to using the embedded English text. The language settings of the browser have no influence on this mechanism.

Alternative languages can be supported by installing the appropriate text resource file (as a file named *CSPres.xml*) in the Gateway's home directory. When the Gateway is started (or restarted) the Gateway loads the text resources found in *CSPres.xml* and the Management forms then appear in the chosen language.

To create a CSPres.xml file, rename the appropriate CSPres_xx.xml file in the Caché bin directory to CSPres.xml.

For example, to convert to Spanish:

1. Rename CSPres_es.xml to CSPres.xml
2. Restart the Web Server. (You must restart because the language text affects the CSP module for the given Web Server.)

To convert back to English (Default):

1. Rename CSPres.xml back to CSPres_es.xml
2. Restart the Web server.

4.1.2 Security Considerations with CSP Web Gateway Management Page

By default, only clients local to the Gateway's hosting computer are allowed access to the CSP Web Gateway Management page. The browser through which the management forms are accessed must be running on the same machine as the web server and CSP Gateway. For example:

http://localhost:<port_no>/csp/bin/Systems/Module.cxw

You can add additional clients to the list of authorized administrators by adding the client IP addresses to the `System_Manager` parameter in the SYSTEM section in CSP.ini (in C:\Intersystems\Cache\CSP\bin). The `System_Manager` parameter represents a comma- (or plus-, +) separated list of clients (by IP address) who may access the CSP Web Gateway Management page. The directive shown below grants access to three remote clients in addition to the default local access.

```
[SYSTEM]
System_Manager=190.8.7.6, 190.8.7.5, 190.8.7.4
```

For new Gateway installations, for which there is no local browser available, manually create this configuration setting by editing CSP.ini.

The *System_Manager* parameter in CSP.ini is equivalent to the **Systems Manager Machines** setting, found under the **Default Parameters** section of the CSP Web Gateway Management page. You can specify wildcard and numeric ranges in the entries for the **Systems Manager Machines** parameter.

The following example indicates that the last part of the IP address can take the value of a number between 4 and 6 inclusive.

```
[SYSTEM]
System_Manager=190.8.7.4-6
```

The previous example is a more convenient way of writing:

```
[SYSTEM]
System_Manager=190.8.7.6, 190.8.7.5, 190.8.7.4
```

You can also use wildcards, such as, in this example:

```
[SYSTEM]
System_Manager=190.8.7.*
```

The following directive grants access to all clients:

```
[SYSTEM]
System_Manager=*.*.*.*
```

However, it is not recommended that such a directive be used on operational systems.

There are shortcomings in using this scheme as a way of protecting the CSP Web Gateway Management page. This scheme does not provide strong security. To check web clients, the IP address of a client is obtained from the CGI environment variable *REMOTE_ADDR*. Client IP addresses can be spoofed.

The use of a proxy between the client and the web server/Gateway installation effectively translates all client IP addresses to that of the proxy. In this scenario you would have to either specify the proxy's IP address as a Gateway Systems Manager (which would effectively grant access to all web users coming in through the proxy) or, preferably, enable the designated systems managers to bypass the proxy layer altogether.

The IP-based scheme, while useful as a first line of defense, should not be relied upon as the sole means through which access to the CSP Web Gateway Management page is controlled – certainly not for CSP installations that are available over the Internet. For production systems it is recommended that you use the hosting web server configuration to control access to the Gateway systems management modules.

4.1.3 Checking System Status

The **System Status** option displays the status of all active CSP connections. You must be a system manager to use this feature. In each of the tables below, click a column head to sort by that column.

First Table: Connections to Caché

The first status table (Connections to Caché) displays information on connections to Caché.

Item	Function
Connection Number	Number that the CSP Gateway assigns to the connection. Your Caché license determines the number of possible connections.
Gateway PID	The Gateway (or hosting web server) process ID for the connection.
Server Name	Name of the Caché system connected to. Mirror members show current configuration name with mirror member name appended.
IP Address	IP address of the Caché system.
TCP Port	TCP port on the Caché server through which the connection communicates. The default port is 1972.
Caché PID	Process ID on the Caché server.
Status	Indicates whether information is being sent to or from the Caché system, as follows: `Free` — no information is being sent and the connection is ready to process the next request. `In Use` — information is being transmitted through the connection. `Private` — the connection is state-aware (preserve mode 1) and not free for general use. `Server` — the connection is being used by the Caché server.
Activity	Number of transactions (hits) the connection has processed.
Close	If available, allows you to forcefully close down the connection by selecting it.

Second Table: Caché Servers

The second status table (Caché Servers) displays information on Caché servers.

Item	Function
Server Name	Name of the Caché system connected to.
IP Address	IP address of the Caché system.
TCP Port	TCP port on the Caché server through which the connection communicates. The default port is 1972.
Total Connections	Number of connections to the Caché system.
Connections In-Use	Number of connections that are currently in use (actively serving a Web request).
Private Connections	Number of connections that are currently in use as state-aware sessions (preserve mode 1).
Total Activity	Number of transactions (hits) the Caché system has processed.

Item	Function
Queued Requests	Number of Web requests that are held in a queue waiting for a free connection to the Caché system. Queued requests are an indication that the Caché license should be increased in order to maintain good performance.

Third Table: Application Paths

The third status table displays information for application paths.

Path Number	The number that the CSP Gateway assigns to the application path.
Path	The application path.
Server Number	The number that the CSP Gateway assigns to the Caché server.
Server Name	The name of the Caché system connected to.
Activity	The number of requests processed by this server for this path since the last Gateway.
Status	The status for this server, one of `Disabled`, `Enabled` or `Offline`. Also the current master (or primary) server in the set is indicated in this column.
Action	If a server is marked as `Offline`, this column contains a button allowing Administrators to mark it `Online/Enabled` again.
Mirror Status	Mirror configuration with Mirror Status of the server, member type (Failover or Async), labelled as Primary as appropriate.

Fourth Table: NSD Internal Status (Where Applicable)

The fourth status table displays internal NSD information. This table only appears for NSD installations or for installations that use the Gateway's response caching facility.

Item	Function
Queued Threads	Where the concurrent load exceeds the number of threads that are available within the Gateway process, this indicates number of requests that are held in a queue waiting for a thread to become free. Under normal operation the value reported should be zero. If a number of queued requests are consistently reported here, consider allocating more processes to the NSD.
Cached Forms	Number of CSP forms cached by the Gateway.
Cached Data (Bytes)	Amount of cached form data held in the Gateway (in Bytes).
Cached Form Activity	Number of hits for forms held in the Gateway cache.
Clear	**Clear Cache** (if available) allows you to manually clear the Gateway form cache. **Clear** (if available) allows you to manually clear the individual forms from the Gateway cache.
Total Cache Blocks	Total number of cache memory blocks available.

Item	Function
Cache Blocks In Use	Total number of cache memory blocks in use.
Refresh	Update status screen.

4.1.3.1 Clearing the Cache

Under certain circumstances, such as during the development process for Web applications, it may be necessary to clear the CSP Gateway cache. To do this:

1. From the Web Gateway Management Main Menu, select **System Status**.
2. On the **System Status** page, there are a number of tables. To clear the cache, in the **Cached Forms** table, select the button in the **Clear** column (the rightmost column) in the **Total** row (the bottom row). (If the **System Status** page does not display a **Cached Forms** table, then there is no currently cached content. This may be because the cache has been cleared recently and nothing has been cached since then.)

This action clears all cached content for the CSP Gateway and removes the **Cached Forms** table from the page until there is new cached content.

Note: If you are attempting to clear the cache during activities beyond the scope of the CSP Gateway Management itself, then, to get to the Web Gateway Management Main Menu, you must first log into the Management Portal and then go to the **Web Gateway Management** page (**System Administration** > **Configuration** > **CSP Gateway Management**).

4.1.4 Closing Connections Manually

If your Caché system shuts down while a CSP connection is still active, CSP continues to try to connect to the system until one of the following occurs:

- It successfully reconnects to the system.
- CSP is shut down.
- The connection is manually closed.

If your Caché system is scheduled for extensive downtime, you may want to close the connections from this option. To close sessions manually:

1. From the Web Gateway Management Main Menu, select **Close Connections**.
2. Select the server from which you wish to disconnect. You can select * (asterisk) to close all connections to all servers.
3. Select **Close Connections**.

Note that you can close the connections while the Caché system is down.

4.1.5 Testing Server Connections

The **Test Server Connection** option on the CSP Web Gateway Management page menu is useful to test CSP Gateway connectivity to your Caché systems. Note that you must be a system manager to use this feature.

To test CSP connectivity:

1. From the Web Gateway Management Main Menu, select **Test Server Connection**.
2. Select the desired Caché system from the displayed list.
3. Select **View Server-side Log** to see the CSP error log on the Caché Server.

4. Select **Connect**.

Depending on your selection and the state of the server connection, you receive one of the following results:

Result	Meaning
CSP Test Form	CSP is installed and working properly. For Caché v5.0 (or earlier) the form lists the variables (both user-defined and CGI Environment) that were passed to the server with the request. The form just shows the basic parameters returned by the target Caché server (version and process ID). The new minimal `connection worked` form was introduced for security reasons.
Server Event Log	You should receive this page if you selected View Server-side Event Log.
System Availability Error	This error occurs any time that Caché is unreachable. If there are no additional error messages, check to ensure your Caché system is running.
Server is currently unavailable	The CSP Gateway may have timed out while trying to establish a new connection.
No response from the Web Server	Check to see if your web server is running. If it is running, shut it down and restart it.
Not Enough Sessions All channels to the server are busy: Please try later	You have reached your connection limit and must close some of the connections you have running.

In all cases where an error condition is returned, check the Event Log for additional and more specific error information. Consider raising the Log Level to capture even more diagnostic information where necessary.

4.1.6 Viewing the Event Log

Use the **View Event Log** option from the Main Menu to read the contents of the Event Log.

By default, each day's log entry, CSP.log, is renamed CSP.log.old. To save all logs in files named with date and time, check Retain All Log Files on the **Default Parameters** page. in the Each log entry is marked with a header record which captures the date, time and additional information with respect to the context in which the log entry was made.

Example:

```
>>> Time: Wed Jul 26 15:40:57 2006; RT Build: 664.896 (win32/isapi);
Log-Level: 9; Thread-Id: 2236; Connection-No: 0; Server: LOCAL;
Server-PID: 3028; Page: GET /csp/samples/menu.csp
```

Select **Clear Log** to clear all current entries from the Event Log.

The Log can be displayed in either ascending date/time order (the default) or descending date/time order. Select the hyperlink at the top right-hand corner of the form to reverse the display order. This hyperlink acts as a toggle between the two modes.

Finally, most browsers are unable to render more than about 1MB of log data in a single form. Therefore, as the volume of log data returned approaches 1MB the Gateway terminates the display and prompt for the next page of data (see the **More** hyperlink at the bottom left-hand corner of the form). Additionally, a **Top** hyperlink is provided at the bottom right-hand corner of the form to allow you to quickly go back to the first form in the series.

4.1.7 Using the HTTP Trace Facility

The HTTP trace facility builds on the Event Log information already captured for log levels v9 (record raw HTTP request) and v9r (record both HTTP request and response). The trace is accessed via the **View HTTP Trace** option.

The trace window consists of two main frames. The left-hand frame contains a list of HTTP requests processed by the Gateway by time and a unique request ID (assigned by the Gateway). As each request is selected, the request and response data is shown in the right-hand frame. Hyperlinks allow easy navigation between the request and response message.

The request listing is automatically refreshed every few seconds (currently 7) so that the overall effect is of a real-time monitor operating in much the same way as a proxy such as TCPTrace.

Note: Note that the HTTP request headers reported by the Gateway are reconstituted because the hosting web server always assumes responsibility for parsing the request headers. The Gateway reassembles the complete header from the CGI environment variables supplied by the web server. However, if a request is passed directly through the NSD component (that is, effectively bypassing the web server), then the request header recorded is byte-for-byte the same as it was when dispatched from the client.

4.1.8 Configuring Default Parameters

The **Default Parameters** option on the CSP Web Management page menu provides you with a mechanism for maintaining the global (system-wide) configuration parameters for the CSP Gateway. Note that you must be a system manager to use this option.

When you configure access to a particular Caché Server, any unspecified optional parameters and/or custom system forms are automatically inherited from the global configuration. For example, if you do not set a **Server Response Timeout** parameter for a specific server, that server inherits the global **Server Response Timeout** setting.

The Default Parameters are made up of up to three components:

1. CSP Gateway
2. Security
3. Connections to Caché
4. ASP Redirect
5. Internal HTTP Server (NSD builds only)
6. Custom Error Messages

4.1.8.1 CSP Gateway

This section contains a parameter that is globally relevant to the whole Gateway installation.

Instance Host Name

This is the network host name for this particular instance of the CSP Gateway. The Gateway generates a default value which is shown beneath the text box. The value of this parameter is transmitted to Caché with the request data as system variable CSPIHN. Caché based software can use the value to access management services provided by the Gateway over the network.

The format for this parameter is: `server_name:port`

Maximum Connections

Maximum number of connections to Caché that can be created from this Gateway instance. Note that prior to 2012.2, the default value in Caché followed the HTTP 1.1 recommendation and was set to 2. In Cache version 2012.2 and later versions, the default value is set to 6 (since most modern browsers, such as Internet Explorer 8 allow 6 connections per session. Increasing this value allows better application responsiveness if an application uses more connections but may also result in heavier server resource utilization.

Changes to the Maximum Connections parameter only take effect after a Gateway (or hosting web server) restart.

Maximum Cache Size

Maximum amount of shared memory to be reserved for the purpose of caching CSP response data.

The cache size may be specified as follows:

In Bytes	n
In Kilobytes	nK
In Megabytes	nM

The default value for this parameter is 250K – which is the amount of memory typically required to cache the static components used in the CSP/Zen samples. This value can be raised or lowered as required.

Changes to the Maximum Cache Size parameter will only take effect after a Gateway (or hosting web server) restart.

Web Server ID Cookie

Suppresses the Web Server ID Cookie (*CSPWSERVERID*). It can be set to:

- Enabled (Default)
- Disabled

The Web Server ID Cookie is used to enable load balancers to implement passive cookie affinity for CSP applications. However, there are situations where it is desirable to suppress the automatic generation of this cookie. For example, in proxy applications where the web request is transparently passed to other servers for processing.

The Web Server ID Cookie is not dispatched when returning resources that are deemed to be static (i.e. images and JS files). In this context, static files will include all responses generated by Cache that are not accompanied by a Web Server ID Cookie. An exception to this rule is made for cases where the application is configured to `Never` use session cookies. In this case the Web Server ID Cookie is included with all responses (as before).

4.1.8.2 Security

If a username and password are defined here, then all system managers must provide this username and password to access the CSP Web Gateway Management page.

If you forget the password, manually edit the Gateway configuration file CSP.ini and remove the *Username* and *Password* parameters from the `SYSTEM` section of this file. Then you can access the CSP Web Gateway Management page without a username and password and enter a new username and password if required.

```
[SYSTEM]
Username=cm
Password=1Bx4tt88mttAWaf7isJg3Urqc2zE
```

You can configure the following CSP security parameters:

Access to these forms

Enable or disable access to the CSP Web Gateway Management page using this option. The default is **Enabled**. When access is **Disabled** you cannot re-enable access using the CSP Web Gateway Management page. To re-enable access, manually edit the configuration file CSP.ini. Set the `SM_Forms` parameter to `Enabled` in the `SYSTEM` section of this file.

```
[SYSTEM]
SM_Forms=Enabled
```

Username

Username required to access the CSP Web Gateway Management page.

Password

Password required to access the CSP Web Gateway Management page.

Password (Confirm)

When the password is modified, confirm the new value here.

Session Timeout

The amount of idle time (in seconds) that an active Systems Management session remains logged on. After this time has expired, the management session expires and the manager is automatically logged out of the CSP Web Gateway Management page.

System Manager Machine/s

Defines a list of client machines (by IP address) through which you can access these Systems Management options. Any client with System Manager access can add or delete access to any CSP system, change any setting in the configuration file, and close down any active sessions. The addresses are separated by either a comma or a plus sign. In this example, two clients have System Manager access:

```
127.0.0.1, 45.123.231.12
```

If this field is undefined, only a client operating on the same machine as the CSP Gateway (that is, the web server host) can configure CSP.

This field is supplemented with a check box (**Override Username and Password**) which, if checked, allows listed client machines to be exempt from entering a username and password to gain access to the Management Forms.

Custom Login Form

Defines a custom login form that controls access the Gateway Management Suite. This parameter can be either the full path to a physical file or a link which enables the hosting web server to serve the form.

Examples:

```
C:\Inetpub\wwwroot\login.html
/login.html
```

If a physical file name is specified then the Gateway retrieves and dispatches the form to the client. Otherwise, it sends an 'HTTP Redirect' response header to enable the client to request the form directly from the hosting web server. The custom form must implement an HTTP POST request to login Gateway Administrators.

The essential form fields are shown below:

```
<FORM METHOD=POST ACTION="/csp/bin/Systems/Module.cxw">
<INPUT TYPE=HIDDEN NAME="CSPSYS" VALUE="17">
<INPUT TYPE=HIDDEN NAME="CSPSYSsmSection" VALUE="SYSTEM">
<INPUT TYPE=TEXT NAME="CSPUNM" SIZE='20' VALUE="">
<INPUT TYPE=PASSWORD NAME="CSPPWD" SIZE='20' VALUE="">
<INPUT TYPE=SUBMIT NAME="CSPSYSbOK" VALUE="Login">
```

Where CSPUNM is the Username and CSPPWD the Password. The text assigned to the Login (submit) button (shown as 'Login' above) can be changed.

A simple but complete example is shown below:

```
<html>
<head>
<title>CSP Gateway Management</title>
</head>
<h2>CSP Gateway Management</h2>
<FORM METHOD=POST ACTION="/csp/bin/Systems/Module.cxw">
<INPUT TYPE=HIDDEN NAME="CSPSYS" VALUE="17">
<INPUT TYPE=HIDDEN NAME="CSPSYSsmSection" VALUE="SYSTEM">
<BR>
Username:
<INPUT TYPE=TEXT NAME="CSPUNM" SIZE='20' VALUE="">
<BR>
Password:
<INPUT TYPE=PASSWORD NAME="CSPPWD" SIZE='20' VALUE="">
<BR>
<INPUT TYPE=SUBMIT NAME="CSPSYSbOK" VALUE="Login">
</form>
</html>
```

4.1.8.3 Connections to Caché

This section contains parameters related to maintaining connections to Caché.

Server Response Timeout

The maximum number of seconds allowed for the target Caché server to respond to a request from the web server. The timeout refers to a period of no activity, so, for example, sending a line of HTML data every second for 10 hours does not cause a timeout. The minimum allowable value for this field is 5 seconds.

The value set here is the default for the system. If an Inherited Value is specified, the value came from the Default Parameters page. You may, however, set a different value on individual server-specific configurations or within the application itself.

Note that if you have an Apache server, you can also set this value using *Timeout* in the Apache http.conf file. The lower of these two values is triggered first.

Queued Request Timeout

This is the maximum number of seconds that a request can remain in a queue waiting for an available connection to the appropriate Caché system. The minimum allowable value is 5 seconds. If an Inherited Value is specified, the value came from the Default Parameters page.

No Activity Timeout

This parameter is relevant to stateless connections only. The parameter indicates the maximum amount of time (in seconds) that a stateless connection remains open in an idle state before closing. If this timeout is exceeded, the session automatically closes. This facility prevents stateless sessions accumulating on your Caché server, particularly after periods of high activity where a large number of connections were opened to cope with the increased workload. If a value is not specified, stateless connections remain open until they are manually closed. If an Inherited Value is specified, the value came from the Default Parameters page.

Apply timeout to all Connections

Applies the **No Activity Timeout** option to all connections (including those making up the minimal connection pool). If this option is not checked, the Gateway does not apply the **No Activity Timeout** to the minimal connection pool (as defined by the **Minimum Server Connections** parameter). If this option is checked the Gateway applies the timeout to all connections in the pool. This option is used by installations that have a very low level of CSP usage and, as a result, have a preference for all CSP processes to time out. If an Inherited Value is specified, the value came from the Default Parameters page.

Event Log Level

Controls what information is written to the Event Log. See the "Event Logging Parameters" section for details.

Event Log File

Specifies a location for the Event Log file (CSP.log). If not specified, it is written to the directory hosting the Gateway installation. For example:

To specify an alternative location for CSP.log:

/opt/logfiles/cspgateway/

To specify an alternative location and file name for the Event Log:

/opt/logfiles/cspgateway/event_log_01012006.log

Retain All Log Files

By default, Caché copies the CSP.log file into a file named CSP.log.old. Subsequent log rotations overwrite CSP.log.old with the current contents of the Event Log. To retain all log files, instead of this daily overwriting, check **Retain All Log Files**. If you enable this option, then the day's log is named with the date and time as: CSP_YYYYMMDD_hhmm.log

Event Log Rotation Size

This defines the size after which log rotation should take place. The default value is blank which effectively means that the Gateway will maintain one log file which will grow until such time as the administrator manually clears it.

If rotation is required, the size may be specified as follows:

In Bytes	n
In Kilobytes	nK
In Megabytes	nM

The minimum size that can be specified is 100K. This value is automatically set if the administrator attempts to set a lower value in the maintenance suite.

Rotated copies of the log file, if retained, are named according to the date and time of rotation as follows:

CSP_YYYYMMDD_hhmm.log

Where:

YYYY	year
MM	month
DD	day of the month
hh	hour of the day
mm	minutes past the hour

For example:

CSP_20090109_1830.log (Log rotated at 18:30 on 9th January 2009)

Rotated log files that are not to be retained are named as: CSP.log.old

In order for this facility to work, the Gateway must have create/write access to the directory hosting the Gateway binaries (i.e. the location where the main log file is kept). If, for whatever reason, the Gateway is unable to perform a successful rotation it will continue writing to the current log file: CSP.log.

This field is supplemented with a check box (**Retain All Log Files**) which, if checked, instructs the Gateway to keep all log files according to the naming scheme outlined above. Otherwise a single rotated file named CSP.log.old is kept.

SSL/TLS Library Path

Specifies the path to the OpenSSL libraries: libssl.so and libcrypto.so. The default position is for the Gateway to source these libraries locally in its home directory. See "Overriding the Library Path If You Use SSL/TLS" in the section "Kerberos Library" in this book for more information.

Preserve Mode Exclude File Types

Allowsr static files to be served asynchronously in state-aware applications. In state-less applications, statics (files other than csp, cls, csr, and zen) are processed asynchronously with respect to the main session. In other words, requests for these files bypass the session lock and can be processed concurrently outside the main processing stream for the application.

This parameter allows this scheme to be extended to state-aware applications. State-aware applications are not only subject to the conventional session lock but are also subject to the connection lock in the CSP Gateway. The connection lock is responsible for ensuring that all requests for the user/session are routed to the same Cache process. For applications that rely on static components being served from Cache, this leads to excessive request queuing which, in turn, can lead to browser instability (such as hangs).

Use this parameter to define a list of (space separated) file types (by extension) to process asynchronously and therefore exempt from connection/session locking in the Gateway and Cache. If the list is prefixed with *- (asterisk hyphen) then all files are processed asynchronously EXCEPT those defined in the following list.

Examples

```
Preserve Mode Exclude File Types=gif jpg jpeg
```

Process files of type GIF, JPG and JPEG asynchronously with respect to the state-aware session:

```
Preserve Mode Exclude File Types=*- csp cls csr zen
```

Process all files asynchronously with respect to the state-aware session EXCEPT those of type CSP, CLS, CSR and ZEN. This, incidentally, is the rule applied in the CSP engine for state-less applications.

This mechanism can be monitored using Gateway Log Level 4 (v4). When invoked for a request, a record similar to the one shown below is added to the log.

```
 >>> Time: Fri Oct 04 14:56:40 2013 ...GET /csp/samples/zenutils.js
     State-Aware Session (preserve == 1)
     Process this request concurrently in the pool of state-less connections (File Type=js)
```

4.1.8.4 ASP Redirect

Web Document Root

This is the full physical path to the document root directory of the web server. For example, for Microsoft IIS Web Servers, this path is usually c:\InetPub\wwwroot. This parameter is only required if you plan to use the facility within CSP to send the CSP output through the Microsoft ASP engine to render the final page.

Temp ASP Directory

This is the full physical path to a directory where the CSP Gateway can temporarily store Microsoft ASP content. This parameter is only required if you plan to use the facility within CSP to send the CSP output through the Microsoft ASP engine to render the final page.

4.1.8.5 Internal HTTP Server

This section is only relevant to the NSD.

This section contains the following parameters:

Service Status

The HTTP server can be either Enabled or Disabled. Select either:

- Enabled
- Disabled

The default is `Enabled`.

In the interests of security, it is best to disable this facility, unless it is intended that the NSD should be able to respond to raw HTTP requests.

NSD Document Root

For cases where the NSD is intended to be used as a stand-alone web server in its own right, this parameter defines the full physical path to the web documents root.

For example:

```
/opt/cspgateway/home/
```

If the server is used to server CSP applications then the broker components should be installed under:

```
/opt/cspgateway/home/broker/
```

The static files used to support the CSP samples:

```
/opt/cspgateway/home/samples/
```

The static files used to support the Management Portal:

```
/opt/cspgateway/home/sys/
```

4.1.8.6 Custom Error Pages

The **Error Pages** section of the global configuration screen allows you to customize CSP Gateway error messages and system responses. These can be set on a global or per-Caché server basis. To customize the default CSP responses, perform the following:

1. From the Web Gateway Management Main Menu, select **Default Parameters**.
2. In the **Error Pages** section, enter the name of the web application page that you wish to replace the corresponding Gateway page with. Enter the full physical path to your web application page, or enter a path relative to that of the CSP Gateway.
3. Select **Save Configuration**.

The CSP Gateway system responses that you can customize are as follows:

Server Error

Page to display when the CSP Gateway encounters an internal error. For example, an error occurs if there is a problem communicating with a Caché server. The specific error is always recorded in the CSP Gateway Event Log.

Server Busy

Page to display when all available CSP connections are in use.

Server Unavailable

Page to display when the Caché server (or application) has been deliberately disabled from within the configuration

Server Timeout

Page to display when the request has timed out.

Connection Closed

Page to display when you log out of a state-aware session.

4.1.8.7 Event Logging Parameters

The **Event Log Level** field allows you to control what information the CSP Gateway writes to the Event Log. Logging options are defined as a string of characters, each character representing a logging command. The value set here for the Event Log Level is the default for the system (that is, all Caché servers); however, you may set a different value for individual Caché Servers.

The CSP Gateway writes the Event Log to the serial file named CSP.log. This file is placed in the same directory as the CSP Gateway runtime module. You can view or clear the Event Log from the CSP Web Management page menu. The logging parameters are used mainly for troubleshooting. The following table shows the logging options, which can be expressed in lower- or uppercase.

Logging Option	Function
E	Record all errors. This option allows you to monitor connection failures.
V	Verbose: Record the basic connection dialog between the CSP Gateway and a Caché system. Use this option to record the strategic points of communication between the CSP Gateway and a Caché server. There are 7 levels to this command (1 to 7). Each successive level records more detailed information. The levels are accumulative. For example, level V3 includes all log information specified for V1 and V2.
EV	Enter EV to turn on basic event logging. The higher log levels generate a large volume of data in the log file and should only be used for diagnosing problems. For production systems it is recommended that the log level should be set to no higher than EV.
V1	Same as V.
V2	In addition to the information specified for previous levels, this level records: • Information regarding basic connection management between the CSP Gateway and Caché (Start and Close points for each connection). • Transmission interrupts received from the browser. • Cases where connections to Caché are forcefully closed (Due to no response from Caché or other errors where the connection can't be recovered). • Access violations in state-aware (preserve mode 1) sessions (For example, Invalid Session ID).
V3	In addition to the information specified for previous levels, this level records: Caché headers and HTTP headers.

Logging Option	Function
V4	In addition to the information specified for previous levels, this level records: Information regarding the serialization of state-aware sessions.
V5	This is reserved for future use.
V6	In addition to the information specified for previous levels, this level records: • Headers to the data blocks sent to Caché. • Request Data from the Web Server (except multipart attachments). • Headers to the data blocks received from Caché.
V7	In addition to the information specified for previous levels, this level records: The full content returned from Caché.
V9	Record incoming HTTP request data. The HTTP headers and posted content (where applicable) are recorded. (Does not record info for levels 1–7.) This log directive can be further extended and refined. • v9r: In addition to logging all HTTP requests, record all HTTP responses. • v9a: Record all HTTP requests to http.log in the Gateway home directory. • v9b: Record all HTTP requests on a per-session basis. Log files of the form http[session_id].log is created in the Gateway home directory, where *session_id* is the 10-Byte session ID. • v9m: Log all multi-part posts in the Gateway home directory. The raw incoming HTTP request are recorded together with the individual components in both their encoded and decoded form.
s	Sessions: Record information about the management of session tokens: • The point at which new session IDs are allocated. • For existing sessions: an indication as to whether the session token was extracted from a cookie or the form/URL variable CSPCHD. • For all requests: the final session ID transmitted to Caché.
c	Connections: Record information about connections made using the Kerberos Library (CCONNECT). Instruct the Gateway to record all CCONNECT functions called, the input parameters, and results. For the sake of brevity, the content of the input and output buffers to and from Caché are not recorded at this level. Set a log level of `C` (upper-case C) to record, in addition to the CCONNECT function calls, the contents of the input and output buffers.
t	Transmission: Record the contents of the raw data buffers as they are dispatched to Caché. Raising the log level further to `T` (uppercase T) results in the raw response buffers being captured too. All non printable characters are recorded in their escaped form.

Logging Option	Function
p[n]	Performance: Instructs Gateway to capture information to assess the performance of the CSP installation. *n* is the number of seconds (total service time) below which data is not recorded for a request. For example, a directive of p records data for all requests, p2 records data for requests taking longer than 2 seconds to service. The following information is recorded. • Total time to service request: The total time spent in servicing the request (from the time it reaches the Gateway to the time at which the last Byte of response data leaves the Gateway environment. • Obtain [NEW] connection to Caché: Time taken between the request reaching the Gateway and a connection to Caché being reserved for the purpose of servicing the request. The message recorded indicates if a new connection is created during this time (as opposed to an existing one being reused). • Send request to Caché: Time taken between the first and last Byte of request data being read from the web server and dispatched to Caché. • Processing request in Caché: Time taken between the last Byte of request data being dispatched to Caché and the first Byte of response data being received by the Gateway. • Receive response from Caché: Time taken between the first and last Byte of response data being received from Caché and dispatched to the web server.

Logging Option	Function
pp[n]	Provides detailed timing information as follows: • Pre-processing of request: Time taken to identify the target Cache server; includes the initial handover from the web server and basic request processing to identify the server. • Obtain [NEW] connection to Cache: Time taken to allocate a connection to the appropriate Cache server. Indicates whether a new connection is created (instead of an existing one reused). • Format request: Time taken to parse and format the request message for transmission to Cache. • Send request to Cache: Time taken between the first and last byte of request data read from the web server and dispatched to Cache. • Processing request in Cache: Time taken between the last byte of request data dispatched to Cache and the first byte of response data received by the Gateway. • Post-processing of response(b): When a content-length header is required, this reports the time taken for the dispatch of the response data back to the client via the web server. • Post-processing of response(c): Time taken between the dispatch of the response and the Gateway being ready to read the response footer data from Cache. The footer data is part of the internal communication protocol between the Gateway and Cache and includes control information (For example: instructions to change the preserve setting for the session). • Receive footers from Cache: Time taken to receive the response footer data from Cache. • Post-processing of footers: Time taken to process footer data and respond to instructions received. • Release connection to Cache: Time taken to release the active connection to Cache. • Cleanup: Time taken to release resources used in servicing the request and return control back to the hosting web server.

4.1.9 Configuring Server Access

The **Server Access** option allows you to:

- Configure CSP Gateway access to named Caché servers.
- Copy the entry of a configured server to another name. This is a quick method of adding a new server.
- Disable access to a configured Caché server.
- Delete a configured server entry.
- Add new servers.

Each Caché system accessed by the CSP Gateway must be defined here. Any unspecified optional parameters or custom system forms are automatically inherited from the CSP Gateway global configuration.

4.1.9.1 Adding a Server Configuration

To configure access to a Caché server:

1. From the Web Gateway Management Main Menu, select **Server Access**.
2. Select **Add Server**. The second configuration screen appears. Note that many parameter fields have default settings.
3. In the **Server Name** text box, enter a unique, descriptive name for the server. This logical name is used to identify the server configuration in the CSP configuration file.
4. Enter the system parameters (described below) for this server configuration.
5. Select **Save Configuration**.

Server Access

The set of base server configuration parameters are as follows:

Server Configuration Parameter	Function
Server Name	Logical name to identify this server configuration in the CSP configuration file.
Service Status	Allows you to enable and disable this configuration (default is Enabled).
IP Address	The IP address (physical or virtual) of the Caché server to connect to.
Superserver TCP Port	The TCP port number on which the Caché server is listening for incoming connections. This is the TCP port number of the Caché superserver which is usually 1972.
Configuration is Mirror Aware	Configures a mirror primary as a server to access mirrored databases. In a failover or disaster recovery, the connection is redirected. By default, not selected. *Note*: If you have configured a mirror VIP, do not configure a mirror aware CSP gateway, which will cause the Gateway to ignore the VIP. Instead, simply configure the CSP Gateway to connect to the VIP like any other client. In general, use of a mirror aware CSP Gateway is the appropriate choice only in unusual circumstances. To configure, enter the IP address of one of the failover members. From this failover member, the Gateway obtains a list of the failover and disaster recovery (DR) async members in the mirror and connects to the current primary based on this list (and not the VIP even if one is configured). The CSP connection fails until a primary is found. Once the connection is established, if the mirror fails over, the Gateway changes the connection to the new primary. If no primary can be found among the failover members, the Gateway attempts to find one among the DR asyncs in the list, which enables it to reestablish the connection when a DR async is promoted to primary in a disaster recovery situation. For details, see "Redirecting Application Connections Following Failover or Disaster Recovery" in the "Mirroring" chapter of the *Caché High Availability Guide*.

Stateless Parameters

The set of parameters relevant to stateless connections are as follows:

Stateless Parameter	Function
Minimum Server Connections	The CSP Gateway implements process affinity. This means that it always attempts to reconnect sessions to the same Caché process that serviced their previous request if possible. This parameter specifies the minimum number of connections that the CSP Gateway should make to the Caché server before starting to share the connections among many clients. The higher this number, the more effective process affinity is.
Maximum Server Connections	This is the absolute maximum number of connections that the Gateway is allowed to make to the Caché server. If concurrent usage exceeds this number, the CSP Gateway starts to queue requests. Requests remain in the queue until a Caché connection becomes available to service the request or the *Queued Request Timeout* is exceeded.
Maximum Connections per Session	This represents the maximum number of connections to Caché that can be concurrently used by an individual session.

Connection Security

Connection Security settings are required by the Gateway to access the Caché server. These parameters are discussed in greater depth in a later section. The set of parameters relevant to connection security are as follows:

Connection Security Parameter	Function
Connection Security Level	Level of security required for connecting to the Caché server. Select one of the options: • Password • Kerberos • Kerberos with Packet Integrity • Kerberos with Encryption • SSL/TLS
Username	Username required by the CSP Gateway for connecting to the Caché server.
Password	Password required by the CSP Gateway for connecting to the Caché server.
Password (Confirm)	When you create a new password, confirm the new password by entering it again.
Product	Product being connected to, either Caché or Ensemble.
Service Principal Name	Service principal name. A Generate button is provided for creating a default name with respect to the target Caché server.
Key Table	Full path to the Key Table file.

SSL/TLS Parameters

The following parameters are relevant only to installations using SSL/TLS to secure connections between the Gateway and Caché.

SSL/TLS Parameter	Function
SSL/TLS Protocol	The version of the SSL/TLS protocol to use. The following options are provided: • SSLv2 • SSLv3 • TLSv1 • TLSv1.1 • TLSv1.2 The default TLS protocol is: `TLSv1+TLSv1.1+TLSv1.2`. The default for CipherList is: `"ALL:!aNULL:!eNULL:!SSLv"`
SSL/TLS Key Type	The type of SSL/TLS key file (based on the algorithm used to generate it). The following options are provided: • DSA — Digital Signature Algorithm • RSA — Rivest, Shamir, and Adelman (inventors of the algorithm) The default is RSA.
Require Peer Certificate Verification	If checked, requires peer certificate verification for this installation.
SSL/TLS Certificate File	The full path to the SSL/TLS certificate file for the Gateway. `Example: C:\InterSystems\certificates\clicert.pem`
SSL/TLS Private Key File	The full path to the private key associated with the Gateway's SSL/TLS certificate. `Example: C:\InterSystems\certificates\clikey.pem`
SSL/TLS CA Certificate File	The full path to the certificate for Certificate Authority (CA) for the Gateway's certificate. `Example: C:\InterSystems\certificates\cacert.pem`
SSL/TLS Private Key Password	The password to the SSL/TLS Private Key.

Optional Parameters

The descriptions of the Optional Parameters are given in the "Configuring Default Parameters" section. If any of these parameters is blank, its value is inherited from the CSP Gateway global configuration described in the section "Connections to Caché".

Error Pages

The Error Pages parameters let you customize the CSP Gateway responses. If not specified, the parameters are inherited from the global configuration. For a description of each parameter, see the "Custom Error Messages" section.

4.1.9.2 Copying a Server Configuration

You can quickly configure a new server by copying the configuration entry of an existing server. Having done this, both configuration entries are identical, except for the server name. You can then edit the second configuration and make changes to it (such as changing the IP address).

This feature is also useful for fine-tuning a configuration. By creating a second (temporary) configuration for a server, you can test parameter changes without worrying about losing the original configuration.

To copy an existing server configuration:

1. From the Web Gateway Management Main Menu, select **Server Access**.
2. At the **Server Access** screen, select an existing server name.
3. Select the **Copy Server** option.
4. Select **Submit**. The second configuration screen appears.
5. In the **Server Name** text box, enter a unique, descriptive name for the new server.
6. Select **Save Configuration**.

4.1.9.3 Disabling Access to a Configured Server

Use this facility to prevent users from accessing a configured Caché server through this Gateway installation.

To disable access to a server:

1. From the Web Gateway Management Main Menu, select **Server Access**.
2. At the **Server Access** screen, select an existing server name.
3. Select the **Edit Server** option.
4. Select **Submit**. The Server configuration screen appears.
5. For the **Server Status** parameter, select **Disabled**.
6. Select **Save Configuration**.

To re-enable access, repeat the procedure and select **Enabled** at Step 5.

4.1.9.4 Deleting a Server Configuration

To delete a configured server:

1. From the Web Gateway Management Main Menu, select **Server Access**.
2. At the **Server Access** screen, select a server name.
3. Select the **Delete Server** option.
4. Select **Submit**.

4.1.10 Configuring Application Access

The Configure Application Access option allows you to:

- Configure the path to your CSP application.
- Copy an application path to another path. This is a quick method of adding a new application path.
- Disable access to an application path.
- Delete an application path.
- Add new application paths.

Each CSP application must have the path to its CSP files configured. The configuration for each path identifies the Caché server responsible for running the application. Optional directives for specifying failover and load-balancing are included in the application path's configuration. The default application path, root, (/) is automatically configured when the CSP Gateway is started for the first time. Inheritance is applied to application paths. For example, if a CSP request asks for a file in /Accounts/Invoices and there is no configuration for /Accounts/Invoices, the CSP Gateway uses the configuration defined for /Accounts. If this is not defined, the configuration for the default path of / is used.

4.1.10.1 Adding an Application Path

To configure the path to an application:

1. On the Web Gateway Management page, select **Application Access**.
2. Select **Add Application**. Note that many parameters have default settings.
3. In the **Application Path** text box enter a unique path for the application. This path is the path which appears in the application URLs.

   Note: A Caché installation creates a new /csp configuration. If you have configured /csp as your application, your configuration is overwritten when you install a new build of Caché. To maintain your application configuration, enter a path other than /csp.

   Any directory under /csp works fine, such as /csp/myapplication, but the path cannot contain any dots (periods). These lead to ambiguity for the CSP Gateway. In this example: /csp/samples/menu.csp/csp/aaa/bbb/ccc.cls, the CSP Gateway could either interpret this as a request for /csp/samples/menu.csp/csp/aaa/bbb/ccc.cls or as a REST request for/csp/samples/menu.csp (where PATH_INFO is /csp/aaa/bbb/ccc.cls). The Gateway, working in the web server environment, has no way of resolving these ambiguities.

   CSP is case-sensitive. Specify your path names consistently when you are configuring CSP.

4. Enter the other configuration path and server parameters (described in the tables below) for this application.
5. When you have finished, select **Save Configuration**. Changes you make to the application configuration take effect as new user sessions are created for that application path. Existing users are unaffected.

Application Path Configuration Parameters

The set of base parameters are as follows:

Parameter	Function
Service Status	Enable and disable access to an application via the application path (default is Enabled).
Web Server Physical Path	Path to the corresponding directory on the web server. This setting is particularly important for Microsoft IIS systems where each path configured must be set up as a virtual directory under the web server configuration. Each virtual directory defined within IIS must have a physical path associated with it. The purpose of this additional configuration procedure for IIS is to allow the paths used by CSP to be defined with execute permissions. The default is for execute (and hence access to CSP) to be denied.
Extra CGI Environment Variables	Comma-separated list of additional CGI Environment Variables to be returned to the Caché environment with each and every request. The commonly-used CGI Environment Variables are automatically sent with each request. Enter the wildcard character (*) to instruct the CSP Gateway to send all Environment Variables supplied by the web server to the Caché server with each request.

Parameter	Function
Process with this class	Process files in this path with the specified class. This allows you to build your own request handlers in CSP. For example, you could build your own WebDAV server by creating a WebDAV class and defining it here. There is a WebLink emulation class which makes use of this facility: %CSP.WebLink.
GZIP Compression	Enable or disable GZIP compression for all CSP pages returned in this path (default is Disabled).
GZIP Minimum File Size	Minimum response size, in bytes, for which GZIP compression is invoked. Default is 500 bytes.
GZIP Exclude File Types	This is a list of file types to be excluded from GZIP compression. Files to be excluded can be listed by MIME type (such as image/jpeg) or by common extension (such as jpeg). By default, these common (natively compressed) image files are excluded: `GZIP Exclude File Types: jpeg gif ico png gz zip mp3 mp4` Separate additional types or extensions with a space.
Response Size Notification	This parameter provides configurable control over the method used by the Gateway to notify clients of the amount of data contained in each response. Web clients typically require some form of response size notification if HTTP KeepAlive connectivity is used. Under these circumstances, the Gateway will default to using chunked transfer encoding, provided HTTP v1.1 is in use. If an earlier HTTP protocol is in force it will buffer the response data received from Cache and generate a content length header instead. Also, in cases where the entire response fits into one output buffer a content length header will be generated instead of using chunked transfer. There are scenarios in which it is desirable to instruct the Gateway to specifically use one method or the other. For example, in cases where HTTP v1.1 is used but some intermediary (such as a proxy) is unable to properly support chunked transfer. Also, while not sending any form of size notification (such as, where the *close connection* event is used as the response terminator) should be supported by all web clients, it is nevertheless recommended as 'good practice' that all responses should be accompanied by some form of size notification. Indeed, some clients require this. The following options are provided: • Chunked Transfer Encoding and Content Length (the default) • Chunked Transfer Encoding • Content Length This parameter is supplemented with a check box to instruct the Gateway to always generate a size notification for all requests regardless of whether or not KeepAlive is used.
KeepAlive	Enable or disable HTTP **KeepAlive** connectivity for this path. Default is `No Action` in which case the **KeepAlive** status is determined by the HTTP response headers for each request.

Parameter	Function
Non-Parsed Headers	Enable or disable **Non-Parsed Headers** protocol for this path. Default is **Enabled** in which case HTTP response headers are streamed directly back to the client. If this property is disabled, the response headers are submitted back to the hosting web server. This gives the web server the opportunity to parse the headers and invoke any output filters that may be indicated (For example the Apache Group's mod_deflate facility).

Server Parameters

You can define a list of Caché servers to use for an application and the purpose for which they are to be used. The default server is the first one specified in the list.

Parameter	Function
Servers	The first server listed is the default Caché server. It is used first. If it fails the other listed servers can be used, depending on the option checked. There are three options: • **Use Load-Balancing and Failover** . If the first server fails, use a server that is configured as either failover or load-balancing. • **Use Failover** . If the first server fails (becomes unavailable), use an alternative. • **Use First Server Only** Use only the first server in the list. Do not use another server under any circumstances.
Servers #:	List of servers. The configuration screen shows only three server slots at any one time, but you can define any number of alternative servers. Each server can be checked as **Enabled** or **Disabled**. Default is always **Enabled** See more information in the "Load-Balancing and Failover" section in this book.

4.1.10.2 Copying an Application Path Configuration

You can quickly configure a new application path by copying the configuration entry of an existing path and editing it.

This feature is also useful for fine-tuning a configuration. By creating a second (temporary) configuration for an application path, you can test parameter changes without worrying about losing the original configuration.

To copy an existing application path configuration:

1. From the Web Gateway Management Main Menu, select **Application Access**.
2. On the Application Access screen, select an existing application path.
3. Select **Copy Application**.
4. Select **Submit**.
5. In the **Application Path** text box, enter a new and unique application path.
6. Select **Save Configuration**. The new application configuration takes effect as new user sessions are created for the new application path. Existing users are unaffected.

4.1.10.3 Disabling Access via an Application Path

Use this facility to prevent users accessing a configured application through this Gateway installation.

To disable access via an application path:

1. From the Web Gateway Management Main Menu, select **Application Access**.
2. At the **Application Access** screen, select an application path.
3. Select **Edit Application**.
4. Select **Submit**. The configuration screen for the application path appears.
5. For the **Application Status** parameter, select **Disabled**.
6. Select **Save Configuration**.

To re-enable access, repeat the procedure and select **Enabled** at Step 5.

4.1.10.4 Deleting an Application Path Configuration

To delete a configured application path:

1. From the Web Gateway Management Main Menu, select **Application Access**.
2. At the **Application Access** screen, select an application path.
3. Select the **Delete Application** option.
4. Select **Submit**.

4.1.11 About CSP Gateway Page

This page displays CSP Web Gateway Management page. This information contains the version of the Caché distribution, the Gateway build number, the version of the hosting web server, the active interface, the name and location of the Gateway configuration file (CSP.ini), and the location of the event log.

Example:

```
Version: 2016.2.1.803.0
Gateway Build: 1602.1573d
Web Server Name: localhost
Web Server Type: Apache Web Server: Apache/2.4.10 (win32)
Cache_Server_Pages-Apache_Module/2016.2.1.803.0-1602.1573d
Active Interface: Apache Module
Configuration: C:InterSystems/HealthShare_2/CSP/bin/CSP.ini
Event Log: C:?InterSystems/HealthShare_2/CSP/bin/CSP.log
```

4.1.11.1 CSP Gateway Build Numbers

The Gateway build number is made up of two numeric components. The first number indicates the version of Caché. The second number is the internal Gateway build number.

For example, the string 1602 refers to version 2016.2.

4.2 CSP Gateway and Security

This section describes how the CSP Gateway relates to Caché security features. For more background on authentication, see the chapter "Authentication" in the *Caché Security Administration Guide*.

Gateway connections to Caché can be protected according to the following levels of security:

1. Minimal connection security (as was the case with early versions of the Gateway and Caché).

2. Simple username- and password-based authentication.
3. Kerberos-based authentication and data protection.
4. SSL/TLS-based authentication and data protection.

Remember that security applied here is solely for the purpose of authenticating the Gateway host to the Caché server. It protects against the unauthorized creation of connections to the CSP engine (`%cspServer`). It does not, however, identify an individual *user* of a CSP application. A user of a CSP application can only be positively identified by whatever user login facility is provided by the application itself. For example, a Systems Manager logging on to the Management Portal can only be identified by the username and password supplied to the Management Portal login form.

The stateless nature of the Web should also be borne in mind. There is no fixed relationship between a Gateway connection to Caché and an individual user of a web application. Many users share the same connection.

Authenticating the CSP Gateway to Caché at connection time is important. If an attacker can impersonate a CSP Gateway, it can redirect traffic through a system under his control (by technical means and/or social engineering) and read and/or modify data at will. This is distinct from authenticating individual users to a CSP application. The CSP Gateway's Caché username and password, Windows network credentials, or UNIX® Kerberos key table should never be used by ordinary users.

4.2.1 Gateway Security Parameters

Maintain the following security parameters using the CSP Gateway Web Management application. Under the **Configuration** section, select **Server Access** and choose to edit, copy, or add a server. The **Connection Security** section has the following settings:

- **Connection Security Level** — Choice of:
  - Password
  - Kerberos
  - Kerberos with packet integrity
  - Kerberos with encryption
  - SSL/TLS
- **Username**
- **Password**
- **Product**— Choice of:
  - Caché
  - Ensemble
- **Service Principal Name**
- **Key Table**

4.2.2 Minimal Connection Security

In Minimal Connection Security, the Connection Security Level is set to **Password** and the **Username** and **Password** fields are left empty.

In this mode, there is a minimal level of security applied to the connection between the Gateway and Caché. This mode of operation is the default scenario if an older version of the Gateway is used (that is, an installation without the additional

security parameters). It is also the mode of operation if a newer Gateway is used to connect to an earlier version of Caché (pre version 5.1).

If this mode of operation, ensure that the CSP service (`%Service_CSP`) together with the username under which it operates (for example, `CSPSystem`) is not expecting any form of authentication.

4.2.3 Simple Username- and Password-based Authentication

In Username- and Password-based Authentication, the Connection Security Level is set to `Password` and the `Username` and `Password` fields are applied.

This is the simplest form of authentication that can be applied between the Gateway and Caché.

It should be remembered that Caché passwords are a weak form of authentication since they must be sent over the network as plain text for authentication in Caché. Network sniffing is easy to do and can be used to reveal these passwords. Passwords used in this configuration option must be held in the Gateway configuration file (CSP.ini) in accordance with the following guidelines.

In all cases, the default username and password used for the CSP Gateway is as follows. The installation process creates the CSPSystem user for this purpose. This user (CSPSystem or any other) should have no expiration date; that is, its Expiration Date property should have a value of 0.

```
Username: CSPSystem
Password: SYS
```

Windows

Passwords are encrypted using functionality provided by Microsoft's Data Protection API (DPAPI). The CSP Web Gateway Management page handles the encryption of passwords.

Occasionally, you need to introduce a password outside the context of the CSP Web Gateway Management page, for example, if the Gateway configuration is set up by custom configuration scripts. In this case, the password should be filed as plain text and the Gateway encrypts it when it is started for the first time.

A form of password encryption is used for Windows because ordinary Windows user accounts are occasionally granted membership in the Administrators Group. This is not recommended practice for production systems but it does happen. Encrypting the password offers a higher level of protection for all Windows installations.

It is not possible to move a CSP.ini file containing encrypted passwords to another computer: The passwords must be reentered and, then, re-encrypted on the new machine. This can be a problem in clustered environments that share a CSP.ini on a shared drive. For information on how to handle this, see the "Caché and Windows Clusters" chapter in the *Caché High Availability Guide*.

UNIX®

Passwords are stored in CSP.ini as plain text. Access to the configuration file should be protected by setting the file owner to be the account from which the Gateway (or hosting web server) operates. The access mode should be set to 600.

4.2.4 Kerberos-based Authentication and Data Protection

In Kerberos-based Authentication and Data Protection, three levels of authentication (and data protection) are provided through the Connection Security Level parameter.

1. Kerberos. This option provides initial authentication only for the connection.
2. Kerberos with Packet Integrity. This option provides initial authentication and guarantees data packet integrity.
3. Kerberos with Encryption. This is the highest level of security and provides initial authentication, guaranteed data packet integrity, and, finally, encryption for all transmitted messages.

4.2.4.1 Kerberos Library

To use any of the Kerberos-based modes, the Gateway must be able to load the InterSystems Kerberos client library:

- Windows DLL:cconnect.dll
- UNIX® Shared Object:cconnect.so

Install the appropriate library in a location specified in the PATH environment variable for the Operating System or at one of the following locations relative to the Gateway installation.

- . (that is, local to the Gateway)
- ./bin
- ../bin
- ../../bin

The Gateway attempts to load the library at the time it is first required. If successful, the following status message is written to the Event Log: CSP Gateway Initialization The CCONNECT library is loaded - Version: 5.3.0.175.0. (This library is used for the optional Kerberos-based security between the Gateway and Caché)

If the Gateway is unable to locate or link to the `cconnect` library, a suitable statement of failure and error message is written to the Event Log.

For Kerberized communications between the Gateway and Caché, the Gateway is the Kerberos client.

The procedure for configuring the Gateway to use Kerberos is in the following two sections — "Windows" and "UNIX".

Overriding the Library Path If You Use SSL/TLS

By default, the Gateway expects dependent security libraries (shared objects) to be installed in its home directory (that is, the directory with the Gateway binaries).

If you use SSL/TLS connectivity between the Gateway and Caché, these libraries include the CCONNECT library and SSL/TLS libraries (libssl.so and libcrypto.so). When the Gateway and CCONNECT libraries, loaded in the web server's process space, load a copy of the SSL/TLS libraries, there is a conflict between different versions of the same libraries that were previously loaded by the hosting web server. To ensure that only one copy of the SSL/TLS libraries are loaded in the web server process space, the Gateway must instruct the CCONNECT library to source the SSL/TLS libraries from the same location as those used by the hosting web server.

To do this, specify a path for the SSL/TLS libraries in the Gateway configuration file (CSP.ini) in the SYSTEM section with the CCONNECT_LIBRARY_PATH parameter. Restart the hosting web server after modifying this parameter. For example:

```
[SYSTEM]
CCONNECT_LIBRARY_PATH=/lib/amd64/
```

4.2.4.2 Windows

Kerberos key tables are not implemented for Windows. Therefore, authentication uses network credentials that are either obtained when the hosting service starts in a named account or from the Trusted Computing Base (TCB) when the hosting service runs in the System Logon Session (that is, as `LOCAL SYSTEM`).

Windows domain accounts use a permanent key derived from a password to acquire a Kerberos Ticket Granting Ticket (TGT) and service ticket for the local machine. The local machine must also have a permanent Kerberos key, shared with the Key Distribution Centre (KDC) component of the domain controller. That key can be used to acquire a TGT and service ticket to authenticate to another Kerberos principal such as Caché.

For practical purposes the Gateway, operating within the context of a Windows-based web server is operating through either the Network Service logon session or the System logon session. The account used must have `Log on as a batch job` rights assigned.

The built-in Network Service logon session has access to the machine's credentials and is designed for services that need network credentials to authenticate to other machines. However, the Network Service logon session is not always present. The System logon session can also be used for the purpose of authenticating the Gateway to Caché.

For IIS installations, and ISAPI extensions in particular, using the Network Service login session is the preferred means through which both databases (local and remote) and remote computers should be accessed.

Gateway Configuration

Set the `Service Principal Name` to that of the target Caché server that the Gateway is connecting to. Leave the `Username`, `Password`, and `Key Table` fields empty.

The client principal name (or client username) is that of the Gateway host. This is the Kerberos name representing the Gateway hosts' network service session:

<computer_name>$

Assign this principal the necessary privileges in the Caché server to allow the Gateway's service to operate.

4.2.4.3 UNIX®

This Operating System supports Kerberos Key Tables. The Gateway configuration is conceptually more straightforward for this system.

Gateway Configuration

Set the `Service Principal Name` to that of the target Caché server that the Gateway is connecting to.

Enter the name of the key table file (including the full path) in the Key Table field.

Set the `Username` field to the name of the appropriate key in the key table file.

Leave the `Password` field empty.

The client principal name (or client username) is that of the Gateway host. This is the name used to identify the key in the Kerberos Key Table. Assign this principal the necessary privileges in the Caché server to allow the Gateway's service to operate.

4.2.5 SSL/TLS-Based Authentication and Data Protection

You can use the SSL/TLS protocol to secure communications between the Gateway and Caché.

In this mode, the SSL/TLS transport, as configured for this host, secures connections to Caché. The **SSL/TLS Configuration Name** field should be set to the appropriate value for the target server. The **Service Principal Name** and **Key Table** fields are not relevant and should be left empty.

For more information on creating SSL/TLS client configurations for Caché systems, see the chapter "Configuring the CSP Gateway to Connect to Caché Using SSL/TLS" in the *Caché Security Administration Guide*. See also the subsection, in this book, "Overriding the Library Path If You Use SSL/TLS" in the section "Kerberos Library" on setting a path for the SSL/TLS libraries.

4.3 CGI Environment Variables

CGI Environment Variables are derived both from the client's HTTP request headers and from the environment in which the web server is operating. The CSP Gateway transmits the common environment variables to Caché with each and every request. If extra environment variables are required by the application, they must be explicitly requested in the CSP Gateway configuration (via the **Extra CGI Environment Variables** setting in the Application Access section of the configuration). On the **[System] > [Configuration]** page, select **CSP Gateway Management** and **Go**. Select **Application Access**

The list of environment variables transmitted is shown in the table below together with a brief description of each. Further documentation can be obtained from standard web text books. See also the section "CgiEnvs Property and CGI Environment Variables" in *Using Caché Server Pages*.

Environment Variable	Value
AUTH_PASSWORD	Value entered in the client's authentication dialog. This variable is available only if Basic authentication is used.
AUTH_TYPE	Contains the authentication method that the server uses to validate users when they attempt to access a protected script.
CONTENT_TYPE	For requests which have attached information, such as HTTP POST and PUT, this is the content type of the data.
GATEWAY_INTERFACE	Revision of the CGI specification to which this server complies. Format: CGI/revision
HTTP_ACCEPT	Value of the Accept request header that contains a list of accepted formats (MIME types). For example: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-excel. The values of the fields for the HTTP_ACCEPT variable are concatenated, and separated by a comma (,).
HTTP_ACCEPT_CHARSET	Comma-delimited list of the character encodings that the client accepts.
HTTP_ACCEPT_LANGUAGE	Contains a string describing the language to use for displaying content (such as en-us).
HTTP_AUTHORIZATION	Contains the Base-64 encoded username, password, scheme and realm sent by the client.
HTTP_COOKIE	Holds the contents of the client's cookie(s).
HTTP_REFERER	Holds a string that contains the URL of the page that referred the request to the current page using an HTML <A> tag. Note that the URL is the one that the user typed into the browser address bar, which may not include the name of a default document. If the page is redirected, HTTP_REFERER is empty.
HTTP_SOAPACTION	SOAPAction HTTP request header field can be used to indicate the intent of the SOAP HTTP request. The value is a URI identifying the intent. SOAP places no restrictions on the format or specificity of the URI or that it is resolvable. An HTTP client MUST use this header field when issuing a SOAP HTTP Request.
HTTP_USER_AGENT	Browser the client is using to send the request. General format: software/version library/version.

Environment Variable	Value
HTTPS	Set to either `On` or `Off` (using word, not numerical value). Set to on if the script is being called through a secure server (that is, using SSL/TLS).
PATH_TRANSLATED	Translated version of PATH_INFO, in which any virtual-to-physical mapping is applied to the path.
REMOTE_ADDR	IP address of the remote host making the request.
REMOTE_HOST	Hostname making the request. If the server does not have this information, it should set REMOTE_ADDR and leave this parameter unset.
REMOTE_IDENT	If the HTTP server supports RFC 931 identification, then this variable is set to the remote username retrieved from the server.
REMOTE_USER	Name of the user as it is derived from the authorization header sent by the client
REQUEST_METHOD	Method with which the request was made. For HTTP, this is `GET`, `HEAD`, `POST`, and so on.
SERVER_NAME	The server's hostname, DNS alias, or IP address as it would appear in self-referencing URLs.
SERVER_PORT	Port number to which the request was sent. For example: 80
SERVER_PORT_SECURE	Set to either 0 or 1. If the request is being handled on the web server's secure port, then it is set to 1. Otherwise, it is set to 0.
SERVER_PROTOCOL	Name and revision of the information protocol that the request came in with. Format: protocol/revision
SERVER_SOFTWARE	Name and version of the web server software responding to the request. Format: name/version.

4.4 HTTP Response Headers

CSP and CSP-based applications usually assume the responsibility for formulating a full HTTP response header. For performance reasons the Gateway traditionally streams the response headers, together with the following content, directly to the client via the web server. This mode of operation is known as the *non-parsed header* (NPH) approach. The Gateway does not grant the hosting web server any control over the response headers by passing them back through the dedicated API functions provided by the server. It is assumed that it is the client that needs to read and interpret the response header directives rather than the web server.

However, this assumption breaks down in cases where it necessary for the web server to interpret the response headers in order to invoke further web server-based functionality implied in the header directives generated by CSP. For example, by invoking output filters to further process the response (compression and encryption utilities etc.). Such output filters are usually found not to work for CSP content returned according to the non-parsed header mode of operation.

A facility exists to instruct the Gateway to explicitly pass the response headers through the hosting web server instead of streaming them directly to the client.

To use this facility, set the following CSP Header Directive:

```
CSP-nph: false
```

This directive must be set in the **OnPreHTTP** method. For example:

```
<script language=Cache method=OnPreHTTP arguments=""
returntype=%Boolean>
Do %response.SetHeader("CSP-nph", "false")
Quit 1 </script>
```

When set to `false`, (the default setting for the Gateway is `true`), the `CSP-nph` directive ensures that the hosting web server is properly notified as to the nature of the response through the response headers returned from the CSP engine. As a result, any further processing can be performed as necessary. This is parsed header mode.

When the CSP Gateway is operating in parsed header mode, the hosting web server interprets the response headers and perhaps add header directives of its own. At the very least it adds a Server header to the response. For example:

```
Server: Apache/2.0.48 (Win32)
```

OR:

```
Server: Microsoft-IIS/5.1
```

OR:

```
Server: Sun-ONE-Web-Server/6.1
```

Note that this facility only applies to the use of Gateway implementations that work directly to web server APIs. In other words: everything other than CGI.

If the Gateway CGI modules are used and this facility is required then you must configure the web server to use the non-NPH versions of the CSP CGI modules. For example, use `CSPcgi` instead of `nph-CSPcgi`. The `nph-` prefix used in the name of a CGI module is the standard way of informing the web server that it is not required to read and interpret the response headers returned by the module: in other words operate in non parsed header mode.

The essential difference between the parsed and non-parsed versions of these modules lies in the way the HTTP response status line is formulated. This is the first line in the header block.

For parsed headers, format the HTTP status line as follows:

`Status:` <*status_code*>

Example:

```
Status: 200 OK
```

For nonparsed headers, format the HTTP status line as follows:

`HTTP/1.1`<*status_code*>

Example:

```
HTTP/1.1 200 OK
```

The CGI modules supplied with the Gateway automatically handle these differences internally. The CSP engine always return a standard HTTP header block (2).

See also the Non-parsed Headers parameter in the “Adding an Application Path” section

4.5 Making a CSP Page the Home Page for the Web Server

This section describes how to set CSP page as the web server's default page (or home page). As an example, the procedure for making the CSP samples menu (/csp/samples/menu.csp) the server's default page is described. The web server's default page is accessed via:

```
http://<web_server>/
```

Note that when a CSP page is served in this way, embedded URLs must be specified in full. If relative URLs are used for embedded hyperlinks, the browser interprets these as relative to the documentation root directory and not the CSP root. For example, taking our samples menu as the home page, the URL to, say, the inspector option should be:

```
http://<web_server>/csp/samples/inspector.csp
```

If relative URLs are used, then the browser incorrectly interprets this link as:

```
http://<web_server>/inspector.csp
```

4.5.1 Internet Information Services

1. Open the Internet Services Manager window.
2. In the left-hand window, navigate to the Default Web Site.
3. Right-click **Default Web Site** and select **Properties** from the menu to display the Default Web Site Properties window.
4. Select the **Documents** tab.
5. Select **Add** to display the Add Default Document Name window.
6. Enter the document name (/csp/samples/menu.csp) for the Default Document Name window.
7. Select **OK** .
8. Ensure that the **Enable Default Document** check box is selected.
9. In the Directories tab, use the arrow keys to move the new default document (/csp/samples/menu.csp) to the top of the list.
10. Select **Apply** and **OK** to save and activate all changes.

The new default page (/csp/samples/menu.csp) must physically exist relative to the web server's documentation root directory. It is only necessary to create an empty file. For example, if your document's root is c:\inetpub\wwwroot proceed as follows:

```
cd c:\inetpub\wwwroot
md csp
cd csp
md samples
cd samples
copy con menu.csp
^Z
```

4.5.2 Sun Web Servers

The procedure described in this section is only available with Gateway build 663.763 (and later).

The following directive specifies the home page for a Sun server in the default section of obj.conf:

```
NameTrans fn="home-page" path="/csp/samples/menu.csp"
```

This directive, however, does not result in the CSP form menu.csp becoming the home page for the server. The reason for this is that the server does not update the environment variables relating to the page requested before transferring control to the CSP Gateway. The Gateway sees the incoming request as a request for / instead of /csp/samples/menu.csp. Netscape-based servers expect NSAPI extensions to accept the responsibility for recognizing this scenario and update the variables identifying the page requested and its path accordingly. You can work around this behavior as follows:

1. Define the CSP home page in the default section of obj.conf:

   ```
   NameTrans fn="home-page" path="/csp/samples/menu.csp"
   ```

2. The section that describes the mapping between CSP files and the Gateway modules must be modified to include the home-page-path directive as follows:

   ```
   <Object ppath="*/*.[Cc][Ss][Pp]">
   Service method=(GET|HEAD|POST) fn=csp_req home-page-path="/cache-install-dir/csp/samples"
   </Object> \
   <Object ppath="*/*.[Cc][Ll][Ss]">
   Service method=(GET|HEAD|POST) fn=csp_req home-page-path="/cache-install-dir/csp/samples"
   </Object> \
   <Object ppath="*/*.[Zz][Ee][Nn]">
   Service method=(GET|HEAD|POST) fn=csp_req home-page-path="/cache-install-dir/csp/samples"
   </Object>
   <Object ppath="*/CSPn3Sys.so">
   Service method=(GET|HEAD|POST) fn=csp_req_sys
   </Object>
   <Object ppath="*/CSPn3.so">
   Service method=(GET|HEAD|POST) fn=csp_req
   </Object>
   ```

It is not entirely necessary to specify the path to the home page in the home-page-path property within the Service directive, but if you do, it results in the PATH_TRANSLATED environment variable taking the value that it would have done had /csp/samples/menu.csp been requested directly. In other words, PATH_TRANSLATED for the home-page (/) is returned as:

/install-dir/csp/samples/inspector.csp

instead of:

/csp/samples/inspector.csp

4.5.3 Apache Servers

Find the `DirectoryIndex` directive in the Apache configuration file. For example:

```
DirectoryIndex index.html index.html.var
```

Add the new default page for the web server at the head of the list:

```
DirectoryIndex /csp/samples/menu.csp index.html index.html.var
```

4.6 Compressing the Response to Requests for CSP Forms (GZIP/ZLIB)

Compressing the response generated by the CSP engine before dispatching it to the client is advantageous because it can dramatically reduce the network bandwidth required to transport the response to the client. From the client's perspective the performance of the application is improved. This is particularly true for clients accessing the application through mobile devices over slower telecommunications networks. There is, of course, a cost in terms of the web server host's CPU time that's required to actually compress the data but this is a small price to pay for the advantages.

The advantage of serving compressed response data is particularly marked for CSP pages for which large volumes of response data are generated.

There are two methods for implementing GZIP in a web server environment.

- Using the Gateway's own interface to the GZIP library described here.
- Using a GZIP output filter as an add-on to the hosting web server.

Most web servers offer add-on facilities for compressing data. Windows/IIS offers a `gzip` filter (implemented as an ISAPI filter). The Apache Group offer a compression filter implemented as an add-on module (mod_deflate.c – which, rather confusingly, implements `gzip` compression not `deflate`). There is also a third-party module for Apache called mod_gzip.c. There are a number of third-party GZIP products available as add-ons for most web servers.

The advantages of implementing a compression solution directly in the CSP Gateway are as follows:

- Ease of setup and configuration.
- Greater flexibility in controlling which CSP files are to be compressed.
- The Gateway receives the response content from Caché in fairly large chunks; therefore the performance of the compression and the degree of compression achieved is better if the data is submitted to the compressor functions in large buffers.
- Finally, it has been discovered that if Chunked Transfer Encoding is enabled at the CSP Gateway level and if the Apache mod_deflate output filter is enabled for the same resources, then Windows browsers are occasionally unable to display the response content.

The Gateway makes use of the freely available GZIP (or `zlib`) library for implementing data compression. The compression algorithm used is described in RFCs (Request for Comments) 1950 to 1952.

4.6.1 Installing the GZIP/ZLIB Library

You can download the GZIP/ZLIB library from the following site:

http://www.zlib.net/

This resource was developed by Jean-loup Gailly and Mark Adler (Copyright (C) 1995-2006).

The library is freely available for all platforms on which the Gateway is supported. It is implemented as a DLL for Windows (zlib.dll) and a shared object (or shared library) for UNIX® systems (libz.so or libz.sl). The library libz.so (or libz.sl) is usually pre-installed on all Linux systems (it is usually installed in /usr/lib/ or /usr/local/lib).

The Gateway dynamically links to the ZLIB library when response compression is requested for the first time. Thereafter the ZLIB library remains loaded until the Gateway is closed down.

For Windows, the ZLIB library should be installed in the Windows System32 directory:

C:\WINDOWS\SYSTEM32\ZLIB.DLL

It should be noted that in the latest distributions, the library is named as ZLIB1.dll. This must be renamed to ZLIB.DLL in order for the Gateway to find it.

For UNIX® systems, the library (libz.so or libz.sl) is usually installed in one of the following locations:

- /usr/lib/
- /usr/local/lib/

If the Gateway is able to load the ZLIB library on demand and identify all the required functions, the following initialization message is written to the Event Log:

```
CSP Gateway Initialization
The ZLIB library is loaded - Version 1.2.3.
       (This library is used for the optional GZIP compression facility)
```

If the Gateway cannot find or link to the ZLIB library, it operates as before (pages are returned without being compressed). A statement of failure is written to the Event Log.

4.6.2 Using the GZIP/ZLIB Library

The Gateway implements two modes of operation (1 and 2) for compressing the response data using the ZLIB library:

1. In this mode, the Gateway streams all data received from Caché into the compressor. When all the data has been processed, the compressor streams the compressed data back to the Gateway at which point it is forwarded on to the client.

   This mode offers the best possible compression at the expense of slightly higher latency. Of course, the latency is more pronounced for larger forms.

2. In this mode, the Gateway streams all data received from Caché into the compressor. On each and every call, the compressor makes as much compressed data as it can available to the Gateway at which point it is forwarded on to the client.

   This mode offers the lowest possible latency at the expense of slightly reduced level of compression. Of course, the reduction in the degree of compression achieved is more pronounced for larger forms. Generally speaking, mode 2 is more appropriate for CSP applications where it's usually not possible to know, in advance, how much data a CSP response contains.

If (and only if) the Gateway is able to successfully compress the data stream returned from Caché, it modifies the HTTP response headers to include the appropriate Content-Encoding directive. For example:

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=ISO-8859-1
Set-Cookie: CSPSESSIONID=000000000002119qMwh3003228403243; path=/csp/samples/;
CACHE-CONTROL: no-cache CONNECTION: Close DATE: Fri, 15 Aug 2003 10:05:18 GMT
EXPIRES: Thu, 29 Oct 1998 17:04:19 GMT PRAGMA: no-cache Content-Encoding: gzip
```

Before attempting to compress response data, the Gateway always checks the value of the `Accept-Encoding` HTTP request header (the HTTP_ACCEPT_ENCODING CGI environment variable). The Gateway only compresses a response if the client has indicated that it is capable of dealing with compressed content.

For example:

```
Accept-Encoding: gzip, deflate
```

There are several methods for specifying that a CSP response should be compressed. These are discussed in the following sections.

4.6.3 Specifying Compression for Individual Pages

The `%response` object contains a property called `GzipOutput`. If this property is set to true (or the mode required) the Gateway attempts to compress the response.

```
<script language=Cache method=OnPreHTTP arguments=""
                returntype=%Boolean>
        Set %response.GzipOutput = 2
        Quit 1
</script>
```

Compression can also be specified on a per-page basis by adding the **CSP-gzip** directive to the HTTP response headers. This must, of course, be done in the `OnPreHTTP` method. For example:

```
<script language=Cache method=OnPreHTTP arguments=""
               returntype=%Boolean>
        Do %response.SetHeader("CSP-gzip", "2")
        Quit 1
</script>
```

The `CSP-gzip` header directive should be set to the compression mode required (1 or 2).

4.6.4 Specifying Compression for All Pages within an Application Path

Compression can be specified on a per-application path basis. This, incidentally, is the most common method for indicating that compression should be used when using a web server output filter (such as `mod_deflate`).

Use the following configuration parameters in the Gateway Application Access section:

Item	Function
GZIP Compression	If Enabled, all CSP output for that path is compressed. Default is Disabled (no compression).
GZIP Minimum File Size	Controls the minimum response size in bytes for which compression is activated. The default is 500 bytes.
GZIP Exclude File Types	List of file types to be excluded from GZIP compression. Files can be listed by MIME type (such as image/jpeg) or by common extension (such as jpeg). By default, these common (natively compressed) image files are excluded: `GZIP Exclude File Types: jpeg gif ico png gz zip mp3 mp4` Separate additional types or extensions with a space.

4.6.5 Monitoring

An Event Log level of V3 instructs the Gateway to record the degree of compression achieved for all CSP responses that were successfully compressed. The size of the compressed data and the original uncompressed data stream is recorded.

For example:

GZIP Compression for `/csp/samples/inspector.csp GZIP Mode=1; Uncompressed Content Size=19042; Compressed Content Size=2499` (13 percent)

4.7 CSP Page Output Caching

Most web developers are familiar with the way web browsers support a client cache of previously requested web pages. Client-side page caching can improve the performance for individual users by allowing previously accessed pages to be retrieved from local storage (memory or local hard drive) rather than as a result of fetching the document from the original server.

CSP Page Output Caching provides the option to maintain a cache of frequently accessed pages within the CSP Gateway. Since the Gateway is a core component residing on the web server, its cache can be shared amongst all users of that CSP installation. For example, if a single user requests a page that is configured to be placed in the Gateway cache, then all subsequent requests for that page can use the cached copy. Cached pages are available to all users. This facility can have a dramatic effect on performance for two reasons: Firstly, from a user's perspective, pages retrieved from the cache are served extremely quickly and, secondly, the Caché system is not involved in the delivery of cached pages so its work load is significantly reduced.

The Page Output Caching facility implemented for CSP is based on the equivalent mechanism provided by Microsoft's ASP.NET product. This choice of design was made in order to reduce the amount of learning involved in getting to grips with this facility. Most developers are familiar with ASP.NET.

Page caching is controlled by settings within the CSP %response object. Two properties are available for controlling which pages should be cached and for how long. The default behavior of CSP is that pages should not be placed in the cache.

4.7.1 %response.Expires Property

The standard `Expires` property controls how long a page should reside in the cache.

For example, if a page is set to expire in one minute, the Gateway removes the page after it has resided in the cache for 60 seconds.

```
%response.Expires=[60 seconds time]
```

The equivalent ASP.NET directive would be:

```
<%@ OutputCache Duration="60"%>
```

4.7.2 %response.VaryByParam Property

This property allows you to control how many cached versions of the page should be created based on name-value pairs sent through HTTP POST/GET. The default value is `None`. `None` implies that only one version of the page is added to the cache, and all HTTP GET/POST parameters are simply ignored. The opposite of the `None` value is *. The asterisk implies that all name-value pairs passed in are to be used to create cached versions of the page. The granularity can be controlled, however, by explicitly naming parameters (multiple parameter names are separated using semicolons).

The use of the `VaryByParam` property is best illustrated by means of an example. Let's say we're building a web application that is capable of displaying the weather forecast for the 50 United States. This application is completely encapsulated in one page: `Weather.csp`.

`Weather.csp` presents the user with a drop-down list of states. A state is selected from the drop-down list and the value of the state is sent back to `Weather.csp`. For example, `State=WA` or `State=TX`. For the sake of simplicity, let's assume that we're using HTTP GET to send the data. Once an item (i.e. `State`) is selected, the request is sent to the server:

```
Weather.csp?State=WA.
```

Let's assume that the forecast is only updated once a day and that there's a significant overhead in generating the forecast.

We could add the following directives to the `%response` object of Weather.csp:

```
%response.Expires=[2 hours time]
%response.VaryByParam="State"
```

The ASP.NET equivalent would be:

```
<%@ OutputCache Duration="10800" VaryByParam="State" %>
```

This would result in every single page, for each state, being cached independently of one another for two hours:

```
Weather.csp
Weather.csp?State=WA
Weather.csp?State=TX
...etc.
```

Now suppose that we add further functionality to show the weather for a specific city. In order to cache pages based on the state and city parameter, we would change the cache directives to:

```
%response.Expires=[2 hours time]
%response.VaryByParam="State;City"
```

The ASP.NET equivalent would be:

```
<%@ OutputCache Duration="10800" VaryByParam="State;City" %>
```

The `VaryByParam` property allows us to cache multiple versions of the same page based on parameters sent through HTTP GET/POST. Be extremely careful when using *, as this can potentially fill the output cache with pages that are not frequently accessed. Remember, the more specific we make the `VaryByParam` property, the more frequently the Gateway can serve pages from the cache. For example, when only specifying a state, we have 51 versions of the page in the cache (50 states + the version with no parameters). When the city parameter is added, and assuming that we have an average of 15 cities per-state, we suddenly increase the number of potentially cached pages to 751.

The Gateway automatically evicts pages from the cache if it becomes constrained by memory as a result of the total volume of the cache becoming too large.

4.7.3 Preserving the User's Session ID for Cached Pages

The requesting user's session ID must be preserved within pages retrieved from the cache, regardless of whether the session is being maintained via a Cookie (`CSPSESSIONID`) or via the token (Form/URL variable: `CSPCHD`).

When a page is cached it is cached against the identity of the user that actually retrieved the page from Caché. This page contains the session ID as either the `CSPSESSIONID` cookie or as the `CSPCHD` token. Before serving a cached page to the user, the Gateway replaces all occurrences of both variables with the requesting user's session token. In fact, the session cookie is actually removed from the cache, which achieves the same thing because the Cookie is preserved on the requesting user's browser.

For example, let's suppose that a page is cached by user `xxxxxxx`. The page is cached with the following identity:

```
Set-Cookie: CSPSESSIONID=xxxxxxx;
```

Now, when another user `aaaaaaa` subsequently retrieves this page from the cache, the cookie is, theoretically, changed to:

```
Set-Cookie: CSPSESSIONID=aaaaaaa;
```

In fact, as mentioned previously, the cookie is simply left alone on the requesting user's browser.

The Gateway must, however, take action for pages that maintain the user's session through the session token, `CSPCHD`. In this case, the initial cached page contains references to the original user as shown below:

```
<A HREF="/csp/page.csp?CSPCHD=xxxxxxx">
<INPUT TYPE=SUBMIT NAME="CSPCHD" VALUE="xxxxxxx">
```

The Gateway automatically changes the value of the session token to reflect the identity of the requesting user. For example, when user `aaaaaaa` subsequently requests this page from the cache, the Gateway modifies these lines as follows:

```
<A HREF="/csp/page.csp?CSPCHD=aaaaaaa">
<INPUT TYPE=SUBMIT NAME="CSPCHD" VALUE="aaaaaaa">
```

4.8 CSP with Microsoft Active Server Pages (ASP) and VBScript

You can mix CSP pages with ASP pages and vice versa provided the integrity of the user's session is maintained for both environments. Both environments maintain their sessions using identifiers stored in cookies. CSP stores its session ID in a cookie named with a prefix of `CSPSESSIONID` and ASP uses a cookie named with a prefix of `ASPSESSIONID`.

For example:

```
CSPSESSIONID=00200001000614eX37yi00363806761900000000000000000000000000
ASPSESSIONIDCCATBQCQ=DPLBEBECJLCJKIHKIKEFCCOA
```

Provided the session ID for each environment is protected when serving pages from the other environment, the application should be able to seamlessly and transparently swap between CSP and ASP based applications.

4.8.1 Client-side VBScript in CSP

VBScript, to be executed in the client's Internet Explorer environment, can be added to CSP pages in the same way that JavaScript is used.

For example:

```
<script language=vbscript>
        document.write "The time is: ",time
</script>
```

4.8.2 Server-side VB-Script in CSP (Serving ASP Content through CSP)

Sometimes you want to construct and serve complete ASP pages from a CSP application. In this case, the pages generated by CSP contain VBScript that is executed on the web server host (in the ASP engine) as opposed to in the client's browser.

The ASP (and ASP.NET) engine, in common with the CSP Gateway, is implemented as an ISAPI extension to IIS. However, there is no mechanism through which the CSP Gateway can route ASP content generated by CSP 'sideways' through the ASP engine.

The facility for serving ASP content through CSP is implemented in the CSP Gateway by saving the ASP page returned by CSP as a physical file under the IIS documents root. Having done this, the client is then redirected to that page. The second redirection step results in the newly created ASP page being processed by the ASP engine, at which point the server-side VBScript is executed before the final page is delivered to the client.

The CSP Gateway manages the redirection to the ASP resource in one of two ways.

- `Server-side redirection` This is the preferred choice. The CSP Gateway, having saved the ASP content returned from CSP as a temporary ASP file, simply requests the page through IIS. IIS ensures that that page is processed by the ASP engine and the content is delivered to the client. The temporary ASP page is then deleted. The advantage of this approach is that, from the client's perspective, it appears as if the ASP content is served directly from CSP. The original CSP page requested is registered in the browser's 'address' box.
- `Client-side redirection` In this scenario, the Gateway redirects the client to the newly-created ASP resource by sending an HTTP redirection header to the client. It is necessary to use this option if SSL/TLS is used between the client and web server. The main disadvantage of this method, apart from the extra round-trip to the client, is that the URL to the temporary ASP page is registered in the browser's address box. This means that the client must be able to re-request the temporary ASP resource (either as a result of users pressing the **Refresh** button or using the back-button against a later page). In order to facilitate this requirement, the Gateway retains temporary ASP files for 24 hours after they have been created.

Temporary ASP files are named according to the following convention:

`CSPASP[n].asp`

Where *n* is a unique number, incremented for each new ASP page created and reset to 1 when the Gateway is restarted.

Web Server and Gateway Configuration

To configure the web server and CSP Gateway to serve ASP pages from a CSP application

1. First, create a physical location for the temporary ASP files under the IIS documents root. For example:

   `C:\Inetpub\wwwroot\asptemp`

2. For many Windows installations (particularly Windows 2000 and later), the default privileges assigned to the web server are not sufficient to allow the CSP Gateway to write to directories under the document root.

   You must, therefore, assign the web server `write` privileges to the asptemp directory, or grant the web server `Administrator` privileges.

   You can modify file-access privileges through Windows Explorer. Alternatively, you can use the following command:

   ```
   cacls C:\Inetpub\wwwroot\asptemp /E /G IUSR_xxx:F
   ```

   Where `IUSR_xxx` is the web server's user authority. The `xxx` component is usually the computer name. You can find the specific name in the Internet Service Manager by navigating to the Authentication methods dialog as follows:

   a. Open the Internet Services Manager.

   b. In the left-hand window, navigate to the Default Web Site.

   c. Right-click the Default Web Site. Select **Properties** from the menu to display the Default Web Site Properties window.

   d. Select the **Directory Security** tab.

   e. Select **Edit** under the Anonymous access and authentication control panel. This displays the **Username** in the Authentication methods dialog.

3. Modify the execute permissions assigned to the asptemp directory such that IIS is able to execute ASP pages in that directory.

   a. Open the Internet Services Manager.

   b. In the left-hand window, navigate to the Default Web Site.

   c. Right-click the Default Web Site. Right-click the temporary ASP directory (asptemp) and select **Properties** from the menu to display the Default Web Site Properties window.

   d. Select the **Home Directory** tab.

   e. Ensure that **Execute Permissions** are set to **Scripts and Executables**.

   f. Select **Apply** and **OK**.

   g. Set the following two parameters in the Gateway configuration (Default Parameters).

      Web Document Root:C:\Inetpub\wwwroot

      Temp ASP Directory:/asptemp

4. Restart IIS.

Usage

To tell the CSP Gateway that the page should be further processed by the ASP engine,

1. Set the `%response.UseASPredirect` property to `true` in the `OnPreHTTP` method of the page.

   ```
   <script language=Cache method=OnPreHTTP arguments=""
              returntype=%Boolean>
           Set %response.UseASPredirect=1 Quit 1
   </script>
   ```

2. Declare VBScript in the CSP page:

   ```
   <%@ LANGUAGE="VBSCRIPT"%>
   ```

Then, you can add both in-line VBScript and complete scripted sections to the page.

Examples:

```
<% Response.Write("<BR><B>ASP inline</B><BR>") %>

<script language=vbscript runat=server>
      Response.Write("<br><b>Message from ASP script</b><br>")
</script>

<%
    Dim n
    For n = 1 To 10 Step 1
    Response.write("<BR>These lines were generated by a 'for' loop in VBSCRIPT: <B>Line # " & n &
"</B>")
    Next
%>
```

Complete example:

```
<script language=Cache method=OnPreHTTP arguments=""
       returntype=%Boolean>
     Set %response.UseASPredirect=1 Quit 1
</script>
<html>
<head>
<title> CSP/VBScript demonstration </title>
</head>
<body>
<h2> CSP/VBScript demonstration page </h2>
<script language=vbscript>
     msgbox "Message from client-side vbscript"
     document.write "The time is ",time
</script>
<% Response.Write("<BR><B>Message from ASP inline</B><BR>") %>
<SCRIPT LANGUAGE=VBScript RUNAT=Server>
     Response.Write("<BR><B>Message from ASP script</B><BR>")
</SCRIPT>
<script language=cache runat=server>
     set x="message from Cach&eacute"
</script>
<br>Message from CSP #(x)# <br>
</body>
9</html>
```

4.9 Implementing HTTP authentication for CSP applications

This enhancement extends the Apache modules (mod_csp*.so/dll and CSPa*[Sys].so/dll) to allow HTTP authentication to be controlled by Caché/CSP.

HTTP authentication of web requests is normally carried out between the web server and client (browser). Because of this it is not usually possible to implement HTTP authentication in custom request handlers hosted by the web server – such as CGI programs and web server API-based request handlers. Of course, such extensions can issue a `401 Authorization Required` response header and, in response to this, the browser displays the HTTP login dialogue. However, in the subsequent request, the web server intercepts the user's login details and attempts to authenticate the user using its own built-in functionality. The username and password does not, at least in the first instance, be passed along to the request handling extension until the web server has authenticated the user on its own terms.

This scheme presents a problem for users of third-party development technologies (such as CSP) who wish to perform HTTP authentication locally (and programmatically) within their technology of choice.

The enhancement described here overcomes these technical difficulties and allows users to perform HTTP authentication in the Caché/CSP environment for Apache-hosted CSP applications. Users of Apache can choose between the three approaches described in the following sections.

4.9.1 Standard HTTP authentication in Apache (mod_auth)

This method is the standard mechanism provided by Apache (through the `mod_auth` module) and is not dependent on the latest modifications to the CSP module. It is mentioned here for the sake of completeness.

As an example, the basic parameters required for protecting the CSP samples using Apache-based authentication are shown in the following configuration block (httpd.conf):

```
<Location "/csp/samples/">
AuthType Basic
AuthName "CSP samples"
AuthUserFile conf/csp.pwd
require valid-user
</Location>
```

Where:

AuthType is the type of authorisation required (usually `Basic`).

AuthName is the realm.

AuthUserFile is the file (relative to the web server root) holding usernames and their associated passwords (in encrypted form). This file is created and maintained by the Apache `htpasswd` utility.

The *require* parameter lists the users who may access the protected resource (the CSP samples in this case). The *valid-user* argument indicates that the user must be defined in the username/password file (as declared in AuthUserFile).

Apache provides for users to be grouped together in user 'groups' – see the *AuthGroupFile* directive for further details:

`http://httpd.apache.org/docs/2.0/mod/mod_auth.html`

4.9.2 Authenticating in CSP at the Same Time as the Request is Processed.

This is the preferred (and best performing) method for implementing HTTP authentication in CSP applications.

The basic parameters required for protecting the CSP samples using CSP-based authentication are shown in the following Apache configuration block (httpd.conf):

```
<Location "/csp/samples/">
AuthType Basic
AuthName "CSP samples"
require valid-user
AuthCSPEnable On
# The following directive is necessary for Apache v2.2 (and later)
AuthBasicAuthoritative Off
</Location>
```

The parameters *AuthType*, *AuthName* and *require* are the standard Apache parameters used for triggering authentication.

The additional *AuthCSPEnable* parameter instructs the CSP module to bypass the authentication checks that would otherwise be performed by Apache (in mod_auth) and pass the user's name and password, along with the original web request, to Caché for authentication. The CSP application must check the user using the following CGI environment variables:

- AUTH_TYPE: This is Basic.
- REMOTE_USER: The user's name.
- AUTH_PASSWORD: The user's password (as plain text).

If the user can be successfully authenticated based on the values held in these parameters then the application should continue and process the request (i.e. return the requested CSP resource). If not, it should return a HTTP `401 Authorization Required` response which, at the very least, should be something like:

```
HTTP/1.1 401 Authorization Required
WWW-Authenticate: Basic realm="CSP samples"
Content-Type: text/html
Connection: close
<html>
<head><title>401 Authorization Required</title>
</head><body> <h1>Authorization Required</h1>
<p> The Cache server could not verify that you are authorized
to access the application. Check your username and password.
</p>
<hr>
</body>
</html>
```

On receiving this message the browser redisplays the login dialogue unless the user has used-up all his/her login attempts (usually 3) in which case the message following the header is displayed instead.

For Caché systems v5.1 (and later), users can implement this method of authentication by modifying the login page. If a request comes in and the user does not have the necessary privileges to run the application then the login page is called, the processing for which can extract the authentication information from the request (such as AUTH_TYPE, REMOTE_USER and AUTH_PASSWORD). If these parameters are correct, the login script can then redirect control to the application page that was originally requested. It should not be necessary to repeat the authentication procedure for all public pages provided the Caché security control layer is deployed.

4.9.3 Authenticating in CSP before the Request is Processed.

This is an alternative method for implementing HTTP authentication in Caché. It is intended primarily for cases where performing authentication at request-processing time in the CSP application would be awkward or time consuming. For example, for existing applications running under Caché v5.0 (or earlier), it might be impractical to add authentication checks to all public pages in the absence of a standard login page.

In this method, the user is authenticated by calling a dedicated authentication class. The CSP Gateway performs this check before dispatching the original request to Caché. When the user's details have been successfully checked by the authentication class, the CSP application need not perform any further Select.

Of course, this method bears the overhead of processing two requests (to Caché) per web request: one for authentication and one for actually dealing with the request for the CSP resource.

The basic parameters required for implementing this method of authentication are shown in the following Apache configuration block (httpd.conf):

```
<Location "/csp/samples/">
AuthType Basic
AuthName "CSP samples"
require valid-user
AuthCSPEnable On
AuthCSPClass /csp/samples/%CSP.HTTPAuthentication.cls
# The following directive is necessary for Apache v2.2 (and later)
AuthBasicAuthoritative Off
</Location>
```

The parameters *AuthType*, *AuthName*, *require* and *AuthCSPEnable* are the same as for method (2).

The additional *AuthCSPClass* parameter defines a class that performs user authentication. The class must extend %CSP.Page and, using the appropriate CGI environment variables, should check the user's login details and return either a `200 OK` response header if the operation is successful or a `401 Authorization Required` response header if not.

A simple authentication class in which user login details are checked against records held in the %Users file is shown below:

```
Class %CSP.HTTPAuthentication Extends %CSP.Page
{
ClassMethod OnPreHTTP() As %Boolean
{
Set %response.ContentType = "text/html"
Set %session.Preserve = 0
Quit 1
}
ClassMethod OnPage() As %Status
```

```
{
Set crlf=$Char(13,10)
Set type=%request.GetCgiEnv("AUTH_TYPE", "")
Set user=%request.GetCgiEnv("REMOTE_USER", "")
Set pwd=%request.GetCgiEnv("AUTH_PASSWORD", "")
Set httpauth=%request.GetCgiEnv("HTTP_AUTHORIZATION", "")
If httpauth'="" {
      Set type=$Piece(httpauth," ",1)
      Set user=$system.Encryption.Base64Decode($Piece(httpauth," ",2))
      Set pwd=$Piece(user,":",2)
      Set user=$Piece(user,":",1)
}
Set auth=0 If $ZConvert(type,"L")'="basic" Set auth=1
If auth=0,user'="",$Get(^%Users(user))=pwd Set auth=1
If auth=1 {
Write "HTTP/1.1 200 OK"_crlf
Write "Content-Type: text/html"_crlf
Write "Content-Length: 0"_crlf
Write "Connection: close"_crlf_crlf
}
Else {
Write "HTTP/1.1 401 Authorization Required"_crlf
Write "WWW-Authenticate: Basic realm=""CSP samples"""_crlf
Write "Content-Type: text/html"_crlf
Write "Content-Length: 0"_crlf
Write "Connection: close"_crlf_crlf
}
Quit $$$OK
}
ClassMethod OnHTTPHeader(ByRef OutputBody As %Boolean) As %Status
{
Quit $$$OK
}
```

For methods (1) and (3) a custom error page can be specified for login failure by using the Apache `ErrorDocument` directive. For example:

```
ErrorDocument /error/my_authentication_error.html
```

Of course, for method (2) the text of the error message is controlled by the CSP application.

4.10 Mirrored Configurations, Failover, and Load Balancing

This section describes:

- Load Balancing and Failover Between Multiple Web Servers
- Load Balancing and Failover Between Multiple Caché Server Instances
- Mirrored Configurations

4.10.1 Load Balancing and Failover Between Multiple Web Servers

In most environments, multiple web servers are used to balance load and provide high availability at the web server layer. A load balancer is typically required to direct user connections to participating web servers. For best performance and resilience, it is recommended that a hardware-based solution is used. A Load Balancing system such as Cisco ACE 4710 or the F5 BigIP LTM appliance is placed in front of a set of web servers. In this configuration, if there are also multiple Caché server instances, such as in a Caché ECP configuration, each web server (and by implication, Gateway instance) should be configured to connect to a specific Caché server instance.

Software based load-balancing and failover systems, though not as robust as hardware based solutions, are much less costly to deploy. Examples of software based solutions include HAProxy and the Apache Group's mod_proxy_balancer. For more information, see the HAProxy site www.haproxy.org

Note: *Important:* Sticky sessions should always be enabled for CSP applications. It is essential that each user session 'sticks' to the same back-end Caché server for the lifetime of the session – unless, of course, a failover event occurs.

Although the above approach is primary recommendation, the Gateway contains a basic (software-based) system for implementing load balancing and failover between multiple Cache servers. In this configuration a Gateway installation is configured to connect to a number of Cache servers. This facility is described in the remainder of this section.

Gateway load balancing and failover is configured in the **Application Access** section of the CSP Gateway Management page (**System Administration** > **Configuration** > **CSP Gateway Management**).

A list of Caché servers may be defined for an application (path). Use the **Servers** parameter to check the purpose for which they are to be used. There are three options:

- **Use Load-Balancing and Fail-Over**
- **Use Fail-Over** Use an alternative only if the main (that is, the first) server becomes unavailable.
- **Use First Server only** Use only the first server in the list.

4.10.2 Load Balancing and Failover Between Multiple Caché Server Instances

In configurations with multiple (equivalent) Caché server instances, such as in a Caché ECP configuration, the Gateway provides a basic (software-based) facility for implementing load balancing and failover between those Caché instances for web applications. An external solution like those described previously is the primary recommendation, however.

The failover mechanism provided by the Gateway is not necessary to implement failover between multiple Caché database servers in a typical High Availability configuration, such as failover clustering or Caché mirroring. Those technologies provide Virtual IP based failover and the Gateway can be configured to connect to that IP address.

The remainder of this section describes the load balancing and failover capabilities provided by the Gateway.

Gateway load balancing and failover is configured in the **Application Access** section of the CSP Gateway Management Page.

A list of Caché servers may be defined for an application (path). Use the options listed under the **Use Alternative Servers For** parameter to select the purpose for which they are to be used. There are two options:

- **Fail-Over**
- **Load-Balancing and Fail-Over**

The default course of action, if neither option is selected, is to use the first Caché server defined in the list.

Following is the list of Caché servers, each designated as *server#* where # is the server number.

The configuration screen shows only three empty server slots at any one time, but it is possible to define any number of alternative servers. Each server can be marked as `Enabled` or `Disabled`. The default setting is `Enabled`.

Load-Balancing is implemented in a round robin fashion. Each new user session is connected to the next available 'alternative' server. Once a user session is established on a server, the Gateway maintains the session on that server unless it becomes unavailable, in which case the session is failed-over to the next available server in the list. State-aware sessions (preserve mode = 1) cannot be failed-over under any circumstances and, consequently, the session is closed if the hosting server becomes unavailable.

4.10.3 Mirrored Configurations

With mirrored Caché configurations, a database is duplicated (or *mirrored*)between participating *mirror members*. A Caché *mirror set* configuration represents the set of participating mirror members for an installation. For a complete description of Caché mirroring, see the chapter Mirroring in the *Caché High Availability Guide*.

If Mirror Virtual IP (or an equivalent technology) is used to provide network redirection to the primary member, then configure the CSP Gateway to connect to that address. No further action is required. The Virtual IP address is always mapped to the mirror primary.

For configurations where the Mirror Virtual IP cannot be used (or does not operate in certain disaster scenarios), it is possible to configure the CSP Gateway to be *mirror-aware*. When the Gateway is mirror-aware, it assumes responsibility for determining which member is primary. To make a Gateway configuration mirror-aware, in the CSP Gateway's **Server Access** section, select **Configuration is Mirror Aware** and provide the address of one of the mirror members.

Note: There are situations where it is not appropriate for a Gateway configuration to be mirror-aware. For example, a Gateway configuration supporting the Management Portal should never be configured to be mirror-aware as the portal must always connect to a specific Caché server regardless of its mirror status.

If a mirror-aware Gateway configuration connects to a Caché server that is not a mirror member then the connection fails and the affected client receives a `Server Availability` error.

The Gateway obtains – from the Member that it first connects to – a list of failover members and disaster recovery (DR) members. The Gateway persists this list in its local configuration file (CSPRT.ini). If the Gateway subsequently cannot connect to the member defined in its configuration then it uses the list previously recorded locally to enable it to identify and connect to alternative members.

The Gateway cycles through the members list until it finds the primary. If it cannot find the primary, the connection fails and the affected client receives a `Server Availability` error.

- The Gateway repeatedly cycles through the list until it finds a member defined as primary.
- To avoid the negative performance impact of a tight looping structure, the Gateway pauses after each cycle for a number of seconds equal to the number of tries.
- For a given HTTP request, the Gateway spends no more time attempting to find the primary than that defined in the **Server Response Timeout** parameter.
- When searching for the primary, the Gateway always connects to failover members first. It only tries async members if it cannot find the primary amongst the failover members. An async member only becomes primary if you manually designate it as primary.

Mirror members appear in the Gateway System Status form when the first connection is made. Mirror members are shown named as the current configuration name (as defined under the Gateway's **Server Access** section) with the mirror set name, mirror, and mirror member name shown as a tooltip.

Two new columns, **Mirror Name** and **Mirror Status** have been added to the 'Caché Servers' table. The name of the mirror set and mirror member are shown in the **Mirror Name** column. The current member status is shown in the **Mirror Status** column: the **Member Type** (`Failover` or `Async`) is shown and the primary member is labelled as `Primary`.

4.11 Process Affinity and State-Aware Mode (Preserve Mode 1)

The architecture of the web is *state-less*. In order to get the best out of web architecture in terms of performance, maintainability and scalability web applications should embrace the state-less paradigm.

By default, CSP applications operate in a state-less environment with respect to the hosting Caché server. The CSP Gateway maintains a pool of connections to Caché and distributes the workload amongst them and increases, within configured limits, (or decreases) the size of the connection pool. Each connection is associated with a single Caché process (as identified by the *$Job* variable).

For a normal CSP application operating in state-less mode, consider the choice of backend Caché process used to serve each request for a client session to be random. The Gateway chooses whichever connection/process happens to be free.

However, in the interests of efficiency, the Gateway does implement a form of Caché *process affinity*. In other words, it attempts, where possible, to route a request for a session to the same Caché process that was used to serve the previous request for that session.

In addition to a measure of process affinity based on session ID, the Gateway also attempts to implement process affinity based on NameSpace. The Gateway keeps track of the NameSpace to which each connection is pointing and delivers, where possible, requests to a connection that is already pointing at the NameSpace required to process the request. This helps in avoiding the overhead incurred in moving resources between different NameSpaces on receiving each web request.

In terms of precedence, session affinity always overrides all other considerations in the selection of a connection. If an incoming request cannot be assigned to the same connection previously used to serve the client session, NameSpace affinity is used instead to influence the final choice.

CSP includes a mode whereby the Gateway routes all requests for a session to a reserved (or private) Caché connection/process. This mode of operation provides a *state-aware* environment with respect to the relationship between CSP sessions and their corresponding Caché processes.

State-aware mode is implemented as CSP Preserve Mode 1

The original motivation for the provision of a state-aware mode of operation was to make it relatively easy to migrate legacy application code from a fixed client-server environment (e.g. terminal applications) to the web. Support for transactions that spanned several HTTP requests was also a consideration in its introduction. However, the limitations outlined in the following paragraphs should be borne in mind when creating state-aware applications.

State-aware applications do not scale as well as their state-less counterparts and it is therefore recommended that new applications (and modifications to existing ones) be designed to be state-less as far as is practically possible. It is recommended that state-aware mode, if used at all, should be applied sparingly in predominantly state-less applications.

Writing complete applications to operate in state-aware mode is not recommended. Apart from the scalability issues that arise as a consequence of the need to reserve a Caché process for each and every session, state-aware applications are unable to take full advantage of modern load balancing and failover solutions because of the very specific requirements for routing requests. Also, state-aware applications are not as fault-tolerant as their state-less counterparts. For example, the recycling of a web server worker process can happen transparently beneath a state-less application but results in all associated state-aware sessions closing. Of course, you can avoid the latter restriction by using the CSP Gateway's NSD component to separate the management of the Gateway process pool from the hosting web server.

Creating a successful state-aware application (or state-aware sections within a predominantly state-less application) requires a certain amount of discipline.

Since all requests for a session must be processed by the same Caché process, a queue must be maintained to serialize access to the private Caché process for cases where the client simultaneously dispatches several requests. The original HTTP v1.1 standard mandated that a client should simultaneously open no more than 2 connections to each server (RFC2616).

However, this limit is configurable and, indeed, the latest generation of web browsers support, by default, up to 8 connections to each server. Needless to say, an increase in the maximum number of connections to each server can have a profound effect on state-aware CSP applications: an application can expect up to 8 requests to be fired concurrently and subsequently held in the queue responsible for controlling access to the single private Caché process.

Another potential pitfall in state-aware mode is the effect of the Server Response Timeout operating between the Gateway and Caché. When the Gateway does not receive a response within the prescribed time limit imposed by the response timeout it has no option but to close the connection with the consequential loss of the state-aware session.

Finally, the effect of client interrupts can cause problems with applications operating in state-aware mode. When a client interrupts a request at (and beyond) the point at which Caché is generating a response, the Gateway attempts to absorb the (now unwanted) response payload in order to retain the connection. If it is unable to do this in a timely fashion it, again, has no option but to interrupt whatever the Caché process is doing by closing the connection and the session is lost. Bear in mind that while the Gateway is attempting to absorb the payload for an interrupted request, further requests for the same session may be arriving and placed in the queue.

In summary, follow the following design goals when creating state-aware applications.

- As far as possible avoid (or use sparingly) client constructs that generate many simultaneous requests (for example: HTML Frameset documents).
- Ensure that responses are generated quickly. This reduces the scope for issues related to timeout and/or client interrupt events. It also relieves pressure on the session queue. If it is envisaged that a task in Caché potentially requires an extended time to complete then consider performing it in another process so that the primary private process can quickly return a response to the Gateway (and client).

4.11.1 Launching State-Aware Mode

Mark a session as *state-aware* by setting the preserve mode as follows:

```
Set %session.Preserve = 1
```

It is recommended that a session be marked as state-aware in the form's OnPreHTTP method:

```
<script language=Cache method=OnPreHTTP arguments="" returntype=%Boolean>
Set %session.Preserve = 1
Quit 1
</script>
```

Issuing the instruction here means that the CSP engine can mark the session cookie (or token) as state-aware before formulating and dispatching the HTTP response headers to the Gateway.

Sessions can be marked as state-aware after the OnPreHTTP method has fired but in this case the session cookie/token has already been formulated. The CSP engine passes the `preserve=1` instruction to the Gateway in the response footers (dispatched after the response payload) and the Gateway marks the connection as `private`and cache the instruction against the session ID so that it can recognize the `unmodified` session token as state-aware when subsequent requests arrive.

If the session is marked as state-aware in the OnPreHTTP method, the Gateway has no need to cache the transition against the session since the information is carried in the session cookie/token which effectively resides on the client.

4.11.2 Maintaining State-Aware Mode and Responding to Errors

Once a session is marked as state-aware and the Gateway has acknowledged the state-transition and marked the connection as *private*, the session transparently operates in state-aware mode until one of the following events occurs:

- The application transitions back to a state-less mode of operation.
- The application programmatically ends the session or the session times-out.

- The private connection closes prematurely as a result of some error condition.

If the private connection hosting a state-aware application is prematurely closed (perhaps as a result of an error condition), the Gateway routes the request to a free state-less connection in the pool and Caché error number 5974 is returned:

```
CSP error occurred
Error: The persistent session is no longer available because the server process does not exist
ErrorNo: 5974
CSP Page: /csp/samples/loop.csp
Namespace: %SYS
Class: <Unknown>
```

At this point the request is operating in state-less mode and it is the application's responsibility to respond to this error: for example, by directing the user back to the login form for the application.

When operating in state-aware mode, the value of %session.NewSession should be checked in every page. Alternatively, the application should check the validity of user specific authentication data stored in %session.Data when the user was first authorized to access the application. These checks are important for security reasons and to ensure that the user session is still securely locked-in to a state-aware mode of operation. An error condition is not automatically raised under these circumstances because it is possible that the session had already (and legitimately) transitioned out of state-aware mode. For example, consider the situation where an incoming session token is still marked as state-aware but the application had already transitioned to state-less mode – this situation arising as a result of a session token being embedded in a form (as CSPCHD) that was served before the transition was made.

Finally bear in mind that when a session is terminated (for example, after it has timed out) the CSP engine deletes all operational data associated with the session, after which point any further incoming requests for that session are treated as though they are for a new session.

The embedded security mechanisms provided by Caché for CSP applications offer protection against the eventualities outlined above. Users are automatically directed to the login form in all cases where a loss of continuity within a state-aware application occurs (with respect to Caché process).

4.11.3 Terminating State-Aware Mode

An application can revert back to a *state-less* mode of operation by setting the preserve mode as follows:

```
Set %session.Preserve = 0
```

It is recommended that this code be executed in the form's OnPreHTTP method:

```
<script language=Cache method=OnPreHTTP arguments="" returntype=%Boolean>
Set %session.Preserve = 0
Quit 1
</script>
```

Issuing the instruction here means that the CSP engine can mark the session cookie (or token) as state-less before formulating and dispatching the HTTP response headers to the Gateway.

A session can be immediately terminated as follows:

```
Set %session.EndSession = 1
```

When you set this property, the session terminates immediately after serving the current request.

You can set a session to timeout as follows:

```
Set %session.AppTimeout = 900
```

The session times out and terminate after the prescribed number of seconds of inactivity. The default is 900 seconds (15 minutes).

4.12 Gateway Registry in Caché

The Caché based CSP Gateway Registry registers each connected Gateway installation with Caché and provides the infrastructure to allow Caché code to interact with those Gateway installations. Such programmatically controlled interactions may include reading and modifying the Gateway's run-time configuration and collecting system status and log information. The relevant classes are as follows:

```
%CSP.Mgr.GatewayRegistry (The Gateway Registry)
%CSP.Mgr.GatewayMgr  (A Connected Gateway)
```

The following code will list all connected (i.e. active) Gateway installations and write the web server IP address, port and Gateway build number to the console window.

```
Set reqistry = $system.CSP.GetGatewayRegistry()
Set gateways = reqistry.GetGatewayMgrs()
For no=1:1:gateways.Count() {
     Set gateway = gateways.GetAt(no)
     Write !,no, " : ",
     Write gateway.IPAddress,":",gateway.Port," ",gateway.Version
}
```

When Caché is first started this list will be empty. As Administrator and User activity increases expect at least two entries to appear: one for the Private Web Server serving the Management Portal and at least one for external web servers supporting applications.

Further documentation will be found associated with the classes listed above. Some code examples follow to illustrate common tasks.

List Default Parameters

```
Kill defaults
Do gateway.GetDefaultParams(.defaults)
ZWrite defaults
```

Update Default Parameter(s)

```
Kill newpars
Set newpars("Server_Response_Timeout")=30
Do gateway.SetDefaultParams(.newpars)
```

List Servers

```
Set status = gateway.GetServers(.servers)
For no=1:1:$ListLength(servers) {
     Set server = $List(servers,no)
     Write !,no, " : ",server
}
```

List Server Parameters

```
Kill serverpars
Do gateway.GetServerParams("LOCAL",.serverpars)
ZWrite serverpars
```

Update Server Parameter(s)

```
Kill newpars
Set newpars("Maximum_Server_Connections")=250
Do gateway.SetServerParams("LOCAL",.newpars)
```

List Application Paths

```
Set status = gateway.GetApplicationPaths(.paths)
For no=1:1:$ListLength(paths) {
     Set path = $List(paths,no)
     Write !,no, " : ",path
}
```

List Application Parameters

```
Kill pathpars
Do gateway.GetApplicationParams("/csp",.pathpars)
ZWrite pathpars
```

Update Application Parameter(s)

```
Kill newpars
Set newpars("GZIP_Compression")="Enabled"
```

Clear Gateway cache

```
Do gateway.ClearCache(“*”)
```

4.12.1 Forcing the Gateway to Reload Its Configuration

There are occasions when the Gateway’s configuration is modified by external agents (i.e. agents other than the Gateway’s own Systems Management Suite).

There are two methods for interactively instructing the Gateway to reload its configuration, and in a way that doesn’t require a complete restart.

4.12.1.1 Using the Caché-Based Gateway Registry

The following Registry Method is provided:

```
Set status = %CSP.Mgr.GatewayMgr.ActivateCSPIni()
```

When successfully called, the Gateway reads its configuration file (*CSP.ini*) and activates all changes made.

4.12.1.2 Using Scripts External to Caché

Scripts should add the following line (case-sensitive) to the SYSTEM section of the modified Gateway configuration file (*CSP.ini*):

```
[SYSTEM]
RELOAD=1
```

The Gateway caretaker daemon checks the *RELOAD* flag approximately every minute and, if correctly set, reloads and reactives its configuration and removes the flag from the file. The following message is written to the Event Log after a successful reload operation:

```
Gateway Management
Gateway Configuration Reloaded and Reactivated
```

4.13 Using WebSockets (RFC 6455)

The web has been built around the request/response paradigm: the client sends a request to the server and the server reacts by sending a response to the client. This paradigm, and HTTP itself, does not allow for an inverted form of this communication protocol whereby a server initiates a request/response cycle with the client. A number of technologies have been developed to create an illusion that a server can initiate a dialogue with a client. These technologies are generally known

as *push-based* or *comet-based* technologies and all suffer from problems that make them unsuitable for general deployment over web infrastructure. The three main techniques in current use are described below.

Short Polling

With this technique a client regularly sends HTTP requests to detect changes in server state and the server is programmed to responds immediately. An empty response signifies no change.

Problems:

- Polling frequency (and responsiveness) is limited by the refresh latency that can be tolerated by the client.
- Each request is a full HTTP request/response round trip which leads to high volumes of HTTP traffic which in turn leads to an unacceptable burden on the server and network infrastructure
- Each message exchange carries the overhead of the HTTP protocol and can be particularly burdensome if the message size exceeds the Maximum Transmission Unit (MTU) which is usually 1500 Bytes for Ethernet.

Long Polling

With this technique a client sends a HTTP request but the server only responds when the client needs to be notified of a change. The client typically sends another “Long Poll” request as soon as the server sends a response message.

Problems:

- Each request is a full HTTP request/response round trip, though this technique involves lower volumes of HTTP traffic than short-polling.
- There is the burden of maintaining persistent connections.
- Each message exchange carries the overhead of the HTTP protocol.
- The success of the technique can be adversely affected by timeouts.

HTTP Streaming

This technique takes advantage of the HTTP protocol’s ability to maintain persistent (or ‘KeepAlive’) connections between the client and server. The client sends an HTTP request which is permanently kept open with the server only responding when the client needs to be notified of a change. The server does not terminate the connection after dispatching a response message and the client waits for the next message from the server (or sends a message of its own to the server).

Problems:

- The whole client/server exchange is framed in a single HTTP request/response round trip and not all servers will support this.
- The success of this technique can be adversely affected by the behavior of intermediaries such as Proxies and Gateways etc. …
- There is no obligation on either side to immediately forward partial responses to the other party.
- The technique can be adversely affected by client buffering schemes.
- The technique can be adversely affected by timeouts.

4.13.1 WebSockets Protocol

The WebSockets protocol (RFC 6455) addresses the fundamental requirement of allowing servers to proactively push messages to clients by providing a full-duplex message-oriented communications channel between a client and its server. The protocol is designed to operate, and hence be secured, over the standard TCP channel already established between the client and server. In other words, the channel already used to support the HTTP protocol between a web browser and web server.

The WebSockets protocol and its API are standardized by the W3C and the client part is included with HTML 5.

Intermediaries (such as proxies and firewalls) are expected to be aware of (and expected to support) the WebSockets protocol.

Browser Support

There have been several iterations in creating the final standard for the WebSockets protocol, each with varying degrees of browser support. The history is summarized below.

- Hixie-75:
  - Chrome 4.0+5.0, Safari 5.0.0
- HyBi-00/Hixie-76:
  - Chrome 6.0-13.0, Safari 5.0.2+5.1, Firefox 4.0 (disabled), Opera 11 (disabled)
- HyBi-07+:
  - Chrome 14.0, Firefox 6.0, IE 9 (via Silverlight extension)
- HyBi-10:
  - Chrome 14.0+15.0, Firefox 7.0+8.0+9.0+10.0, IE 10 (via Windows 8 developer preview)
- HyBi-17/RFC 6455
  - Chrome 16
  - Safari 6
  - Firefox 11
  - Opera 12.10/Opera Mobile 12.1
  - IE 10

The final highlighted section is the most significant for the purpose of developing portable web applications.

Server Support

The server-oriented JavaScript-based Node.js technology arguably offers the most sophisticated, and currently most mature, implementation of the WebSockets protocol. And for this reason, WebSockets have been closely associated with Node.js up until the time of writing (March 2013). However, other web server technologies are quickly catching up and the latest versions of all major web servers now offer WebSockets support as shown below.

- Node.js
  - All versions
- Apache v2.2
- IIS v8.0
  - Windows 8 and Windows Server 2012
- Nginx v1.3
- Lighttpd

The highlighted section is the most significant for the purpose of developing portable web applications with CSP.

Protocol in Detail

Creating a WebSocket involves an ordered exchange of messages between the client and the server. First, the WebSocket handshake must take place. The handshake is based on, and resembles, an HTTP message exchange so that it can pass without problem through existing HTTP infrastructure.

- Client sends handshake request for a WebSocket connection.
- Server sends handshake response (if it is able to).

The web server recognizes the conventional HTTP header structure in the handshake request message and sends a similarly constructed response message to the client indicating that it supports the WebSocket protocol - assuming it is able to. If both parties agree then the channel is switched from HTTP (http://) to the WebSockets protocol (ws://).

- When the protocol is successfully switched, the channel allows full duplex communication between the client and server.
- The data framing for individual messages is minimal.

Typical WebSocket Handshake Message from Client

```
GET /csp/user/MyApp.MyWebSocketServer.cls HTTP/1.1
Host: localhost
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: x3JJHMbDL1EzLkh9GBhXDw==
Sec-WebSocket-Protocol: chat
Sec-WebSocket-Version: 13
Origin: http://localhost
```

Typical WebSocket Handshake Message from Server

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: HSmrc0sMlYUkAGmm5OPpG2HaGWk=
Sec-WebSocket-Protocol: chat
```

Note how the client handshake message requests that the protocol be upgraded from HTTP to WebSocket. Note also the exchange of unique keys between the client (Sec-WebSocket-Key) and server (Sec-WebSocket-Accept).

4.13.2 WebSockets Client Code (JavaScript)

In the browser environment the client side of the WebSocket protocol is implemented in JavaScript code. Standard text books describe the usage model in detail. This document will briefly describe the basics.

Create a WebSocket

The first parameter represents the URL identifying the server end of the WebSocket application. The second parameter is optional, and if present, specifies the sub-protocol that the server must support for the WebSocket connection to be successful.

```
var ws = new WebSocket(url, [protocol]);
```

Example:

```
ws = new WebSocket(((window.location.protocol == "https:")
     ? "wss:" : "ws:") \
     + "//" + window.location.host
     + /csp/user/MyApp.MyWebSocketServer.cls);
```

Note how the protocol is defined as either `ws` or `wss` depending on whether or not the underlying transport is secured using SSL/TLS.

The read-only attribute `ws.readyState` defines the state of the connection. It can take one of the following values:

- 0 The connection is not yet established.

- 1 The connection is established and communication is possible.
- 2 The connection is subject to the closing handshake.
- 3 The connection is closed or could not be opened.

The read-only attribute `ws.bufferedAmount` defines the number of bytes of UTF-8 text that have been queued using the **send()** method.

WebSocket Events

The following events are available.

- `ws.onopen` Fires when the socket connection is established.
- `ws.onmessage` Fires when the client receives data from the server.

Data received in event.data.

- `ws.onerror` Fires when an error occurs in the communication.
- `ws.onclose` Fires when the connection is closed.

WebSocket Methods

The following methods are available.

- `ws.send(data)` Transmit data to the client.
- `ws.close()` Close the connection.

4.13.3 WebSockets Server Code (CSP)

The base Caché class for implementing WebSocket Servers is%CSP.WebSocket

When the client requests a WebSocket connection, the initial HTTP request (the initial handshake message) instructs the CSP engine to initialize the application's WebSocket server. The WebSocket server is the class named in the requesting URL. For example, if your WebSocket server is called `MyApp.MyWebSocketServer` and is designed to operate in the USER NameSpace then the URL used to request the WebSocket connection is:

```
/csp/user/MyApp.MyWebSocketServer.cls
```

WebSocket Events

The implementation of the WebSocket server is derived from the base *%CSP.WebSocket* class. There are three key methods to implement as responses to the following events. Note that the CSP session is unlocked before calling any of these methods.

OnPreServer (optional)

Use this method to invoke code that should be executed before the WebSocket server is established. Changes to the SharedConnection property must be made here.

Server (Mandatory)

The WebSocket server. This is the server-side implementation of the WebSocket application. Messages can be exchanged with the client using the **Read()** and **Write()** methods. Use the **EndServer()** method to gracefully close the WebSocket from the server end.

OnPostServer (optional)

Use this method to invoke code that should be executed after the WebSocket server has closed.

WebSocket Methods

The following methods are provided

```
Method Read(ByRef len As %Integer = 32656,
     ByRef sc As %Status,
     timeout As %Integer = 86400) As %String
```

This method reads up to `len` characters from the client. If the call is successful the status (`sc`) is returned as `$$$OK`, otherwise one of the following error codes is returned:

- `$$$CSPWebSocketTimeout` The Read method has timed-out.
- `$$$CSPWebSocketClosed` The client has terminated the WebSocket.

```
Method Write(data As %String) As %Status
```

This method writes data to the client.

```
Method EndServer() As %Status
```

This method gracefully ends the WebSocket server by closing the connection with the client.

```
Method OpenServer(WebSocketID As %String = "") As %Status
```

This method opens an existing WebSocket Server. Only a WebSocket operating asynchronously (`SharedConnection=1`) can be accessed using this method.

WebSocket Properties

The following properties are provided:

SharedConnection (default: 0)

This property determines whether the communication between the client and WebSocket server should be over a dedicated Gateway connection or asynchronously over a pool of shared connections. This property must be set in the **OnPreServer()** method and may be set as follows:

- `SharedConnection=0` The WebSocket server communicates synchronously with the client via a dedicated Gateway connection. In this mode of operation the hosting connection is effectively 'private' to the application's WebSocket Server.
- `SharedConnection=1` The WebSocket server communicates asynchronously with the client via a pool of shared Gateway connections.

WebSocketID

This property represents the unique identity of the WebSocket.

SessionId

This property represents the hosting CSP Session ID against which the WebSocket was created.

BinaryData

This property instrucst the Gateway to bypass functionality that would otherwise interpret the transmitted data stream as UTF-8 encoded text and set the appropriate binary data fields in the WebSocket frame header.

This should be set to 1 before writing a stream of binary data to the client. For example:

```
Set ..BinaryData = 1
```

4.13.4 WebSockets Server Example

The following simple WebSocket server class accepts an incoming connection from a client and simply echo back data received.

The timeout is set to 10 seconds and each time the **Read()** method times-out a message is written the client. This illustrates one of the key concepts underpinning WebSockets: initiating a message exchange with the client from the server.

Finally, the WebSocket closes gracefully if the client (i.e. user) sends the string `exit`.

```
Method OnPreServer() As %Status
{
   Quit $$$OK
}

Method Server() As %Status
{
   Set timeout=10
   For  {
      Set len=32656
      Set data=..Read(.len, .status, timeout)
      If $$$ISERR(status) {
If $$$GETERRORCODE(status) = $$$CSPWebSocketClosed {
Quit
}
         If $$$GETERRORCODE(status) = $$$CSPWebSocketTimeout {
               Set status=..Write("Server timed-out at "_$Horolog)
         }
      }
      else {
         If data="exit" Quit
         Set status=..Write(data)
      }
   }
   Set status=..EndServer()
   Quit $$$OK
}

Method OnPostServer() As %Status
{
   Quit $$$OK
}

}
```

4.13.5 WebSockets Server Asynchronous Operation

The example given in the previous section illustrates a WebSocket server operating synchronously with the client over a dedicated Caché connection. When such a connection is established it is labeled as `WebSocket` in the status column of the Gateways Systems Status form. With this mode the WebSocket is operating within the security context of the hosting CSP session and all properties associated with that session can be easily accessed.

With the asynchronous mode of operation (`SharedConnection=1`), the hosting connection is released as soon as the WebSocket Object is created and subsequent dialogue with the client is over the pool of shared connections: messages from the client arrive via the conventional pool of Gateway connections to Caché and messages to the client are dispatched over the pool of Server connections that have been established between the Gateway and Caché.

In asynchronous mode, the WebSocket Server becomes detached from the main CSP session: the SessionId property holds the value of the hosting Session ID but an instance of the session object is not automatically created.

The example given previously can be run asynchronously simply by setting the SharedConnection property in the OnPre-Server() method. However, it is not necessary to have a Caché process permanently associated with the WebSocket. The **Server()** method can exit (and the hosting process halt) without closing the WebSocket. Provided the WebSocketID has been retained, the WebSocket can be subsequently opened in a different Caché process and communication with the client resumed.

Example:

```
Class MyApp.MyWebSocketServer Extends %CSP.WebSocket
{

Method OnPreServer() As %Status
{
MYAPP.SAVE(..WebSocketID)
   Set ..SharedConnection = 1
   Quit $$$OK
}

Method Server() As %Status
{
   Quit $$$OK
}

Method OnPostServer() As %Status
{
   Quit $$$OK
}

}
```

Note that the WebSocketID is retained for subsequent use in the **OnPreServer()** method. Note also, the setting of the SharedConnection property in the **OnPreServer()** method and that the **Server()** method simply exits.

Subsequently retrieving the WebSocketID:

```
Set WebSocketID = MYAPP.RETRIEVE()
```

Re-establishing a link with the client:

```
Set ws=##class(%CSP.WebSocketTest).%New()
Set %status = ws.OpenServer(WebSocketID)
```

Reading from and writing to the client:

```
Set %status=ws.Write(message)
Set data=ws.Read(.len, .%status, timeout)
```

Finally, closing the WebSocket from the server side:

```
Set %status=ws.EndServer()
```

4.14 Option for Automated Deployment Sites (Such As Cloud)

This build provides an option to relocate Gateway-maintained version/timestamp parameters from the Gateway configuration file (CSP.ini) to the run-time parameters file (CSPRT.ini). CSPRT.ini is owned and maintained by the CSP Gateway and does not contain configuration settings.

This option satisfies the needs of automated software deployment environments, such as those used to create and maintain cloud-based installations. For these environments, configuration files are maintained outside of their operating environment. When changes are made, redeployment or update operations are triggered. Therefore it is essential that configuration files are not changed (by the CSP Gateway) at run-time.

To invoke this option, set the READONLY parameter in the CSP.ini file before initializing the CSP Gateway (by starting or restarting the hosting web server).

```
[SYSTEM]
READONLY=1
```

The affected parameters are:

```
Configuration_Initialized
Configuration_Initialized_Build
Configuration_Modified
Configuration_Modified_Build
```

Examples:

```
Configuration_Initialized        = Thu Nov 12 18:02:57 2015
Configuration_Initialized_Build  = 1601.1551
Configuration_Modified           = Thu Jul 14 16:47:40 2016
Configuration_Modified_Build     = 1603.1599
```

A

Alternative Configurations for Microsoft Windows

This section describes how to set up atypical configurations for Microsoft Windows. Everyone should read the first sections to see if they apply to your configuration. Then select one of the configuration options. This section contains the following sections:

- Using the Network Service Daemon (NSD)
- Installing the ISAPI and CGI Services
- Alternative Options for IIS7 or Later
- Alternative Options for IIS6 or Earlier
- Alternative Options for Windows Apache

These atypical configurations may require some of the following modules. Refer to the sections describing each option to see which are actually required.

- CSPcms.dll (ISAPI/Native module client to the NSD – if supplied)
- CSPcgi.exe (Runtime module)
- nph-CSPcgi.exe (Copy of CSPcgi)
- CSPcgiSys.exe (Systems Management module)
- nph-CSPcgiSys.exe (Copy of CSPcgiSys)
- CSPmsf1.dll (ISAPI filter – if supplied)

These modules are placed in the following common location:

C:\Inetpub\CSPGateway

The configuration file (CSP.INI) and Event Log (CSP.LOG) are written in this directory for non NSD-based connectivity options.

A.1 Using the Network Service Daemon (NSD)

A.1.1 When to Use the NSD

Several of the options described in this book use the NSD. There are two situations in which you might choose to use the NSD to separate the CSP Gateway from the web server so that you can manage the CSP Gateway independently of the Web Server. These are:

- If your web server distributes its load over multiple server processes, an instance of the CSP Gateway is then attached to each web server process.
- If you have a very large web server installation for which CSP is only a small part; for example, a web server that serves php, static content, .NET, and .ASP applications, as well as CSP applications.

A.1.2 NSD Module Install Locations

If the you use the NSD Module in Microsoft Windows, you install the following two utilities:

- CSPnsd.exe
- CSPnsdSv.exe

On an IIS installation, these are installed in this location:

C:\Inetpub\CSPGateway\nsd

On an Apache installation, these are installed in this location:

C:\Program Files\Apache Group\Apache\CSPGateway\nsd

Run the NSD from within its home directory, C:\Inetpub\CSPGateway\nsd. The configuration file (CSP.INI) and Event Log (CSP.LOG) are written in this directory for NSD-based connectivity options.

A.1.3 Operating the NSD

Use the following procedure to start the NSD.

1. Change to the NSD home directory, such as:

   C:\Inetpub\CSPGateway\nsd

2. Start the NSD with:

   ```
   CSPnsd
   ```

   The NSD starts as a Windows service (`CSPnsd.Sv.exe`). Once registered as a service, you can manage the NSD entirely through the Windows Service Manager.

3. Close down the NSD, by issuing the following command:

   ```
   CSPnsd -stop
   ```

Alternatively, you can enter:

```
CSPnsd
```

This shows the status of the NSD's Windows Service and allows you to perform one of the following actions:

- Stop the NSD service if it is running.
- Continue the NSD service if it is paused.
- Remove the NSD service from the services database.

Alternatively, you can use the Windows Service Manager to manage the NSD. The NSD can be identified in the Service Manager by the description:

```
Cache Server Pages – Network Service Daemon
```

All errors are reported in the CSP event log (that is, CSP.log). This file is created and maintained in the *install-dir*\csp directory. The CSP configuration file CSP.ini also resides in this directory.

Other Startup Options

1. Display help information.

   ```
   CSPnsd -h
   ```

2. Run the NSD interactively in a command window as opposed to as a Windows service. This mode of operation must be used if multiple instances of the NSD are to be run.

   ```
   CSPnsd -v
   ```

3. Give permission to others to run the NSD. Administrators of the NSD (CSPnsd) component can give permission to a group or others to start/stop the NSD using `CSPnsd –m=`*s* where *s* is a startup option.

   s can be one of

   - u for the current user (default)
   - g for the current group
   - o for others
   - a for everyone (`m=ugo`)

   Example: `CPSnsd -m=ug` gives permissions to the group (the Administrator group) to run the NSD. This command gives the CPSnsd.pid permissions of: `-rw-rw---`

   When the command to stop the CSPnsd is issued, it tries to signal the CSPnsd parent process to shut down as before. If this is not possible because the service was started by a different user, a flag is written to the CSPnsd.ini file and the service gracefully closes itself down when it acknowledges this flag. This process takes up to 20 seconds to complete.

A.1.3.1 Starting NSD on Alternative TCP Port

By default, the NSD listens for incoming requests on TCP port 7038. You can override this by starting the service as follows:

`CSPnsd –v` [*port_no*]

Or:

`CSPnsd –v -p`[*port_no*]

- where *port_no* is the TCP port number of your choice.

On startup, the NSD creates the following file:

CSPnsd.ini

Typically, this file contains the following lines:

```
[SYSTEM]
Ip_Address=127.0.0.1
TCP_Port=7038
```

In this context, the clients are the CSP modules contained within, or dynamically linked to, the web server and/or the CSP CGI modules invoked by the server. It is, therefore, essential that this file is not deleted or moved. It is also important that the web server processes can read this file. Set the privileges accordingly, bearing in mind the Windows user under which your web server is operating. The NSD clients attempt to find this file in a location contained within the Windows PATH variable. For example:

\Windows

The CSPnsd.ini file must be moved to this location before starting the web server

Clearly, storing the NSD port number in the CSPnsd.ini file is inappropriate for situations in which multiple instances of the NSD are running. For Apache servers, there is a much better mechanism for communicating the TCP port number of the NSD to its clients.

Set the following environment variables in the Apache configuration to indicate the address and port of the target NSD installation. The values specified in these environment variables take precedence over any values found in the CSPnsd.ini file.

CSP_NSD_NAME — This is the IP address of the NSD. Only use this parameter if the NSD is operating on a remote computer.

CSP_NSD_PORT — This is the TCP port of the NSD.

Example 1:

Distribute the load for two Apache virtual hosts (say, 123.123.1.1 and 123.123.1.2) between two independent NSD installations (listening on TCP port 7038 and 7039).

Add the following directives to the Apache configuration (httpd.conf):

```
<VirtualHost 123.123.1.1>
ServerName 123.123.1.1
SetEnv CSP_NSD_PORT 7038
</VirtualHost>
<VirtualHost 123.123.1.2>
ServerName 123.123.1.2
SetEnv CSP_NSD_PORT 7039
</VirtualHost>
```

Example 2:

Distribute the load for two CSP applications (say, /csp1 and /csp2) between two independent NSD installations (listening on TCP port 7038 and 7039).

1. Add the following directives to the Apache configuration (httpd.conf):

   ```
   <Location /csp1>
   SetEnv CSP_NSD_PORT 7038
   </Location>
   <Location /csp2>
   SetEnv CSP_NSD_PORT 7039
   </Location>
   ```

2. Restart Apache after making changes to its configuration.

In cases where multiple instances of the NSD are running, it is recommended that the separate instances be installed in separate directories, each maintaining its own copy of the configuration and log files (CSP.ini and CSP.log). The CSP Web Gateway Management page for each instance can easily be accessed by using the NSD internal HTTP server. For example:

```
http://localhost:7038/csp/bin/Systems/Module.cxw
```

```
http://localhost:7039/csp/bin/Systems/Module.cxw
```

A.2 Alternative Options for IIS7 or Later

This appendix contains instructions for configuring atypical options for IIS7. To configure one of these options:

1. Follow the steps in the section Summary of Configuration Steps
2. Install ISAPI and GCI services, as described in the section Installing the ISAPI and CGI Services
3. Select one of the following 4 options and follow the directions in that section:
   a. Alternative Option 1: Using the ISAPI Modules (CSPms*.dll)
   b. Alternative Option 2: Using a Native Module with the NSD (CSPcms.dll)
   c. Alternative Option 3: Using an ISAPI Module with the NSD (CSPcms.dll)
   d. Alternative Option 4: Using the CGI Modules with the NSD (nph-CSPcgi*.exe)

A.2.1 Installing the ISAPI and CGI Services

IIS 7 does not, by default, run **ISAPI extension**, **ISAPI filters**, or **CGI modules**. For all the atypical options for IIS7, you must install these services.

Note that, with the **ISAPI extensions** service installed, all versions of the CSP Gateway that have ever been built (even those shipped with Caché v4) work with IIS 7.

Install these legacy services through the Windows Control Panel.

1. Open the Windows Control Panel.
2. Select **Programs and Features** and select **Turn Windows Features on or off**.
3. Navigate to **Internet Information Services** and expand **World Wide Web Services** and **Application Development Features**.

   Select **ISAPI Extensions**. Also select **ISAPI Filters** and **CGI**, if these additional services are required. Select **OK**
4. In the Windows **Control Panel**, open **Administrative Tools** and **Internet Information Services (IIS) Manager**.
5. In the left panel, highlight **[MACHINE_NAME] ([machine_name]\[user_name])**
6. In the middle panel, double-click the **Modules** icon.
7. In the right panel, select **Add Native Module**.
8. In the left panel, expand the top level, expand **Web Sites** and expand **Default Web Site**

```
[MACHINE_NAME] ([machine_name]\[user_name])
              Web Sites
                     Default Web Site
```

9. In the middle panel, double-click **Handler Mappings**.
10. In the middle panel, highlight the **ISAPI-dll** handler.
11. In the right panel, select **Edit Handler Permissions**.
12. Select **Execute** and select **OK**. This allows ISAPI extensions to be invoked through direct calls to the name of the ISAPI DLL.

A.2.2 Alternative Option 1: Using the ISAPI Modules (CSPms*.dll)

Use this option if your CSP Gateway DLLs are unable to support the Native Module interface (the Recommended Option). This is the default (and best performing) solution that was supplied for earlier versions of IIS.

IIS 7 does not, by default, run **ISAPI extensions**, **ISAPI filters** or **CGI modules**. This option requires the **ISAPI extensions** service.

Follow the instructions in the section Installing the ISAPI and CGI Services (If Required) for installing and configuring the ISAPI extensions service.

The web server should be configured such that it recognizes CSP requests (files of type .csp, .cls, and .zen) and passes them to the CSP Gateway for processing.

A.2.2.1 Enabling the ISAPI Extensions

DLLs: CSPms.dll and CSPmsSys.dll

Before these extensions can be used they must be registered with IIS as being "Allowed" applications. This is done in the **Internet Information Services (IIS) Manager** control panel.

1. Open the **Internet Information Services (IIS) Manager** window.
2. In the left panel, highlight**[MACHINE_NAME] ([machine_name]\[user_name])**
3. In the middle panel, double-click **ISAPI and CGI Restrictions**.
4. In the right panel, select **Add**.
5. In the **Add ISAPI or CGI Restriction** dialogue, enter the following details:

   ISAPI or CGI Path: C:\inetpub\CSPGateway\CSPms.dll

   Description: CSPGatewayRunTime

   Allow extension path to execute: Select

   Select **OK**

A.2.2.2 Mapping the CSP File Extensions

Choose *one* of the following configuration methods:

1. Serve all content (including static content) from Cache. Map * to the CSP Gateway. Follow the file map procedure in the section "Registering Additional File Types with CSP" in this book.
2. Serve static content from the web server. Map *only* files of type .csp, .cls, .zen, .cxw to the CSP Gateway.

If you are serving static files from the web server, map the CSP file extensions to the CSP Gateway ISAPI extensions as follows:

Extension	Binary
*.csp	C:\inetpub\CSPGateway\CSPms.dll
*.cls	C:\inetpub\CSPGateway\CSPms.dll
*.zen	C:\inetpub\CSPGateway\CSPms.dll
*.cxw	C:\inetpub\CSPGateway\CSPms.dll

1. Open the **Internet Information Services (IIS) Manager** window.

2. In the left panel expand the top level to reveal the **Web Sites** section, then the **Default Web Site** section. Highlight the **Default Web Site** section:

```
[MACHINE_NAME] ([machine_name]\[user_name])
            Web Sites
                    Default Web Site
```

   Note: This activates CSP for the whole web site. To restrict the use of CSP to specific virtual sub-directories (such as /csp/) focus control on the appropriate subdirectory (under **Default Web Site**) before creating the mappings. Repeat the process for each virtual subdirectory from which CSP content is to be served.

3. In the middle panel, double-click the **Handler Mappings** icon.
4. In the right panel, select **Add Script Map**.
5. In the **Add Script Map** dialogue, enter:

   Request Path: *.csp

   Executable: C:\inetpub\CSPGateway\CSPms.dll

   Name: CSPGateway_csp

6. Select **Request Restrictions**.

   Clear: **Invoke handler only if request is mapped to**

   Select **OK** to return to **Add Script Map** dialogue.

   Select **OK**.

7. At this point you may be prompted as follows:

   “Would you like to enable this ISAPI extension? If yes, we add your extension as an “Allowed” entry in the ISAPI and CGI Restrictions list. If the extension already exists we allow it.”

   Select **Yes**.

   You can later find the list of allowed applications as follows:

   In the left panel, highlight:

   [MACHINE_NAME] ([machine_name]\[user_name])

   In the middle panel, double-click **ISAPI and CGI Restrictions**.

   If the Gateway ISAPI components are not included in the list of allowed applications then add them (as you would have done for IIS 6):

   You can add text of your own choice in the **Description** field. For example:

   `CSPGatewayManagement for CSPmsSys.dll`

   `CSPGatewayRunTime for CSPms.dll`

8. Repeat the above process: Use the **Add Script Map** dialogue to enter the following two mappings:

   Request Path: *.cls

   Executable: C:\inetpub\CSPGateway\CSPms.dll

   Name: CSPGateway_cls

   Request Path: *.zen

   Executable: C:\inetpub\CSPGateway\CSPms.dll

   Name: CSPGateway_zen

Request Path: *.cxw

Executable: C:\inetpub\CSPGateway\CSPms.dll

Name: CSPGatewayManagement

A.2.2.3 Operating and Managing the Gateway

To access the CSP Gateway's systems management suite, point your browser at one of the following locations:

http://<ip_address>/csp/bin/Systems/Module.cxw

http://<ip_address>/csp/bin/CSPmsSys.dll

The CSP engine is automatically invoked for requested files that contain a .csp, .cls, or .zen extension. For example:

http://<ip_address>/csp/samples/menu.csp

If you see an unauthorized user error message, refer to the section "CSP Gateway and Security."

A.2.3 Alternative Option 2: Using a Native Module with the NSD (CSPcms.dll)

IIS 7 does not, by default, run **ISAPI extensions**, **ISAPI filters**, or **CGI modules**. This option requires the **CGI modules** service for running the Gateway Management module (nph-CSPcgiSys.exe).

Follow the instructions in the section for installing the CGI service, Installing the ISAPI and CGI Services (If Required.

Configure the web server so that it recognizes CSP requests (files of type .csp, .cls, and .zen) and passes them to the CSP Gateway for processing.

A.2.3.1 Registering the Runtime Native Module

DLL: CSPcms.dll

Before this module can be used it must be registered with IIS. This is done in the **Internet Information Services (IIS) Manager** control panel.

1. Open the **Internet Information Services (IIS) Manager** window.
2. In the left panel, highlight:

   [MACHINE_NAME] ([machine_name]\[user_name])
3. In the middle panel, double-click the **Modules** icon.
4. In the right panel, select **Add Native Module**.
5. Select **Register** and enter the following details in the **Register Native Module** dialogue:

   Name: CSPcms

   Path: C:\inetpub\CSPGateway\CSPcms.dll

   Select **OK**.
6. In the left panel expand the top level to reveal the **Web Sites** section, then the **Default Web Site** section. Highlight the **Default Web Site** section:

```
[MACHINE_NAME] ([machine_name]\[user_name])
            Web Sites
                    Default Web Site
```

7. In the right panel, select **Add Native Module**.
8. In the **Add Native Module** dialogue select **CSPcms** then select **OK**.

A.2.3.2 Enabling the CGI module for Gateway Management

Executable: nph-CSPcgiSys.exe

Before this module can be used it must be registered with IIS as being an Allowed application. This is done in the **Internet Information Services (IIS) Manager** control panel.

1. Open the **Internet Information Services (IIS) Manager**.
2. In the left panel, highlight:

   [MACHINE_NAME] ([machine_name]\[user_name])
3. In the middle panel, double-click the **ISAPI and CGI Restrictions** icon.
4. In the right panel, select **Add**.
5. In the **Add ISAPI or CGI Restriction** dialogue, enter:

   ISAPI or CGI Path: C:\inetpub\CSPGateway\nph-CSPcgiSys.exe

   Description: CSPGatewayManagement

   Allow extension path to execute: Select

   Select **OK**.

A.2.3.3 Mapping the CSP File Extensions

Note: Do NOT use `Add Wildcard Script Mapping` utility for this file extension mapping process; it will give you an error! Instead, use the utility called `Add Module Mapping for *`.

Choose *one* of the following configuration methods:

1. Serve all content (including static content) from Cache. Map * to the CSP Gateway. If you are configuring CSP so that the Cache server serves all static files, then follow the file map procedure in the section "Registering Additional File Types with CSP" in this book.
2. Serve static content from the web server.

   Map *only* files of type .csp, .cls, .zen, .cxw to the CSP Gateway.

If you are serving static files from the web server, map the CSP file extensions to the CSP Gateway Modules as follows:

Extension	Native Module	Binary
*.csp	CSPms	C:\inetpub\CSPGateway\CSPms.dll
*.cls	CSPms	C:\inetpub\CSPGateway\CSPms.dll
*.zen	CSPms	C:\inetpub\CSPGateway\CSPms.dll
*.cxw		C:\inetpub\CSPGateway\nph-CSPcgiSys.exe

1. Open the **Internet Information Services (IIS) Manager** window.
2. In the left panel, expand the top level to reveal the **Web Sites** section, then the **Default Web Site** section. Highlight the **Default Web Site** section:

```
[MACHINE_NAME] ([machine_name]\[user_name])
            Web Sites
                    Default Web Site
```

Note: This activates CSP for the whole web site. To restrict the use of CSP to specific virtual sub-directories (such as /csp/) focus control on the appropriate subdirectory (under **Default Web Site**) before creating the mappings. Repeat the process for each virtual subdirectory from which CSP content is to be served.

3. In the middle panel, double-click the **Handler Mappings** icon.
4. In the right panel, select **Add Module Mapping**.
5. In the **Add Module Mappings** dialogue, enter:

   Request Path: *.csp

   Module: Select CSPcms

   Name: CSPGateway_csp
6. Select **Request Restrictions**.

   Clear: **Invoke handler only if request is mapped to**

   Select **OK** to return to the **Add Module Mappings** dialogue.

   Select **OK**.
7. Repeat the above process to add the following Module Mappings:

   Request Path: *.cls

   Module: Select **CSPcms**

   Name: CSPGateway_cls

   and

   Request Path: *.zen

   Module: Select **CSPcms**

   Name: CSPGateway_zen
8. In the left panel, highlight the **Default Web Site** section:

```
[MACHINE_NAME] ([machine_name]\[user_name])
            Web Sites
                    Default Web Site
```

9. In the middle panel, double-click the **Handler Mappings** icon.
10. In the right panel, select **Add Script Map**.
11. In the **Add Script Map** dialogue, enter:

    Request Path: *.cxw

    Executable: C:\inetpub\CSPGateway\nph-CSPcgiSys.exe

    Name: CSPGatewayManagement
12. Select **Request Restrictions**.

    clear: **Invoke handler only if request is mapped to**

    Select **OK** to return to the **Add Script Map** dialogue.

    Select **OK**.
13. You may be prompted as follows: “Would you like to enable this ISAPI extension? If yes, we add your extension as an “Allowed” entry in the ISAPI and CGI Restrictions list. If the extension already exists we allow it.”

Select **Yes**.

You can later find the list of allowed applications as follows:

In the left panel, highlight:

[MACHINE_NAME] ([machine_name]\[user_name])

In the center panel, double-click the **ISAPI and CGI Restrictions** icon.

If the Gateway Management CGI module is not included in the list of allowed applications, add it (as you would have done for IIS 6):

You can add text of your own choice in the **Description** field. For example:

```
CSPGatewayManagement for nph-CSPcgiSys.exe
```

A.2.3.4 Operating and Managing the Gateway

This connectivity option depends on the CSP Gateway's Network Service Daemon (NSD).

Start the CSP NSD as described in the section, *Starting the NSD*.

Although CSP pages are served through the higher-performing module (CSPcms.dll), the Gateway's management suite is accessed through the CGI module dedicated to this purpose (nph-CSPcgiSys.exe).

To access the CSP Gateway's Systems Management suite, point your browser at one of the following locations:

http://<ip_address>/csp/bin/Systems/Module.cxw

http://<ip_address>/csp-bin/nph-CSPcgiSys

The CSP engine is automatically invoked for requested files that contain a .csp, .cls, or .zen extension. For example:

http://<ip_address>/csp/samples/menu.csp

If you see an unauthorized user error message, refer to the section *CSP Gateway and Security*.

A.2.4 Alternative Option 3: Using an ISAPI Module with the NSD (CSPcms.dll)

Use this option if your CSP Gateway DLLs are unable to support the Native Module interface (Alternative Option 2).

IIS 7 does not, by default, run **ISAPI extensions**, **ISAPI filters** or **CGI modules**. This option requires both the **ISAPI extensions** and the **CGI modules** service.

Follow the instructions in the section for installing the CGI service, Installing the ISAPI and CGI Services (If Required.

The web server should be configured such that it recognizes CSP requests (files of type .csp, .cls, and .zen) and passes them to the CSP Gateway for processing.

A.2.4.1 Enabling the Runtime ISAPI Extension

DLLs: CSPcms.dll

Before this extension can be used it must be registered with IIS as being "Allowed" applications. This is done in the **Internet Information Services (IIS) Manager** control panel.

1. Open the **Internet Information Services (IIS) Manager** window.
2. In the left panel, highlight: **[MACHINE_NAME] ([machine_name]\[user_name])**
3. In the middle panel, double-click the **ISAPI and CGI Restrictions** icon.
4. In the right panel, select **Add**.

5. In the **Add ISAPI or CGI Restriction** dialogue, enter:

   ISAPI or CGI Path: C:\inetpub\CSPGateway\CSPcms.dll

   Description: CSPGatewayRunTime

   Allow extension path to execute: Select

   Select **OK**

A.2.4.2 Enabling the CGI module for Gateway Management

Executable: nph-CSPcgiSys.exe

Before this module can be used it must be registered with IIS as being an "Allowed" application. This is done in the **Internet Information Services (IIS) Manager** control panel.

1. Open the **Internet Information Services (IIS) Manager** window.
2. In the left panel, highlight: **[MACHINE_NAME] ([machine_name]\[user_name])**
3. In the middle panel, double-click the **ISAPI and CGI Restrictions** icon.
4. In the right panel, select **Add**.
5. In the **Add ISAPI or CGI Restriction** dialogue, enter:

   ISAPI or CGI Path: C:\inetpub\CSPGateway\nph-CSPcgiSys.exe

   Description: CSPGatewayManagement

   Allow extension path to execute: Select

   Select **OK**.

A.2.4.3 Mapping the CSP File Extensions

Choose *one* of the following configuration methods:

1. Serve all content (including static content) from Cache. Map * to the CSP Gateway. If you are configuring CSP so that the Cache server serves all static files, then follow the file map procedure in the section "Registering Additional File Types with CSP" in this book.
2. Serve static content from the web server.

   Map *only* files of type .csp, .cls, .zen, .cxw to the CSP Gateway.

If you are serving static files from the web server, map the CSP file extensions to the CSP Gateway Modules as follows:

Extension	Binary
*.csp	C:\inetpub\CSPGateway\CSPcms.dll
*.cls	C:\inetpub\CSPGateway\CSPcms.dll
*.zen	C:\inetpub\CSPGateway\CSPcms.dll
*.cxw	C:\inetpub\CSPGateway\nph-CSPcgiSys.exe

1. Open the **Internet Information Services (IIS) Manager** window.
2. In the left panel, expand the top level and expand**Web Sites**. Highlight **Default Web Site** .:

```
[MACHINE_NAME] ([machine_name]\[user_name])
            Web Sites
                    Default Web Site
```

Note: This activates CSP for the whole web site. To restrict the use of CSP to specific virtual sub-directories (such as /csp/) focus control on the appropriate subdirectory (under **Default Web Site**) before creating the mappings. Repeat the process for each virtual subdirectory from which CSP content is to be served.

3. In the middle panel, double-click **Handler Mappings**.
4. In the right panel, select **Add Script Map**.
5. In the **Add Script Map** dialogue, enter:

   Request Path: *.csp

   Executable: C:\inetpub\CSPGateway\CSPcms.dll

   Name: CSPGateway_csp
6. Select **Request Restrictions**.

   clear: **Invoke handler only if request is mapped to**

   Select **OK** to return to the 'Add Script Map' dialogue.

   Select **OK**.
7. At this point you may be prompted as follows:

   "Would you like to enable this ISAPI extension? If yes, we add your extension as an "Allowed" entry in the ISAPI and CGI Restrictions list. If the extension already exists we allow it."

   Select **Yes**.

   You can later find the list of allowed applications as follows:

   In the left panel, highlight:

   [MACHINE_NAME] ([machine_name]\[user_name])

   In the middle panel, double-click the **ISAPI and CGI Restrictions** icon.

   If the Gateway ISAPI module is not included in the list of allowed applications then it should be added (as you would have done for IIS 6):

   You can add text of your own choice in the **Description** field. For example:

   `CSPGatewayRunTime for CSPcms.dll`

   `CSPGatewayManagement for nph-CSPcgiSys.exe`
8. Repeat the above process: Use the **Add Script Map** dialogue to enter the following two mappings:

   Request Path: *.cls

   Executable: C:\inetpub\CSPGateway\CSPcms.dll

   Name: CSPGateway_cls

   Request Path: *.zen

   Executable: C:\inetpub\CSPGateway\CSPcms.dll

   Name: CSPGateway_zen

   Request Path: *.cxw

   Executable: C:\inetpub\CSPGateway\nph-CSPcgiSys.exe

Name: CSPGatewayManagement

A.2.4.4 Operating and Managing the Gateway

This connectivity option depends on the CSP Gateway's Network Service Daemon (NSD).

1. Start the CSP NSD as described in the section dedicated to this service.

Although CSP pages are served through the higher-performing ISAPI module (CSPcms.dll), the Gateway's management suite is accessed through the CGI module dedicated to this purpose (nph-CSPcgiSys.exe).

To access the CSP Gateway's Systems Management suite, point your browser at one of the following locations:

http://<ip_address>/csp/bin/Systems/Module.cxw

http://<ip_address>/csp-bin/nph-CSPcgiSys

The CSP engine is automatically invoked for requested files that contain a .csp, .cls, or .zen extension. For example:

http://<ip_address>/csp/samples/menu.csp

If you see an unauthorized user error message, refer to the section "CSP Gateway and Security."

A.2.5 Alternative Option 4: Using the CGI Modules with the NSD (nph-CSPcgi*.exe)

In most cases, the all-inclusive Native Module-based solution (the Recommended Option) is the option of choice, and is the implementation that gives the best performance. The CGI/NSD hybrid is useful for cases where it is necessary, for operational reasons, to manage the Gateway independently of the hosting web server. For example, if multiple instances of the web server are to share the same Gateway installation. In option 1 each instance of the core web server process binds to its own instance of the Gateway.

Another factor in choosing this approach might be that the in-house requirements of your web master (or ISP) dictate that all web server extensions are implemented using the CGI protocol.

IIS 7 does not, by default, run **ISAPI extensions**, **ISAPI filters** or **CGI modules**. This option requires the **CGI modules** service.

Follow the instructions in the section for installing the CGI service, Installing the ISAPI and CGI Services (If Required.

Configure the web server so that it recognizes CSP requests (files of type .csp, .cls, and .zen) and passes them to the CSP Gateway for processing.

A.2.5.1 Enabling the CGI Modules

Executables: nph-CSPcgi.exe and nph-CSPmsSys.exe

Before these modules can be used they must be registered with IIS as being "Allowed" applications. This is done in the **Internet Information Services (IIS) Manager** control panel.

1. Open the **Internet Information Services (IIS) Manager** window.
2. In the left panel, highlight:

   [MACHINE_NAME] ([machine_name]\[user_name])
3. In the middle panel, double-click the **ISAPI and CGI Restrictions** icon.
4. In the right panel, select **Add**.
5. In the **Add ISAPI or CGI Restriction** dialogue, enter:

   ISAPI or CGI Path: C:\inetpub\CSPGateway\nph-CSPcgi.exe

Description: CSPGatewayRunTime

Allow extension path to execute: Select

Select **OK**.

6. Repeat the above steps for nph-CSPcgiSys.exe, entering the following details in the **Restrictions** dialogue:

   ISAPI or CGI Path: C:\inetpub\CSPGateway\nph-CSPcgiSys.exe

   Description: CSPGatewayManagement

   Allow extension path to execute: Select

A.2.5.2 Mapping the CSP File Extensions

Choose *one* of the following configuration methods:

1. Serve all content (including static content) from Cache. Map * to the CSP Gateway. If you are configuring CSP so that the Cache server serves all static files, then follow the file map procedure in the section "Registering Additional File Types with CSP" in this book.
2. Serve static content from the web server.

   Map *only* files of type .csp, .cls, .zen, .cxw to the CSP Gateway.

If you are serving static files from the web server, map the CSP file extensions to the CSP Gateway CGI Modules as follows:

Extension	Binary
*.csp	C:\inetpub\CSPGateway\nph-CSPcgi.exe
*.cls	C:\inetpub\CSPGateway\nph-CSPcgi.exe
*.zen	C:\inetpub\CSPGateway\nph-CSPcgi.exe
*.cxw	C:\inetpub\CSPGateway\nph-CSPcgiSys.exe

1. Open the **Internet Information Services (IIS) Manager** window.
2. In the left panel expand the top level to reveal the **Web Sites** section, then the **Default Web Site** section. Highlight the **Default Web Site** section:

```
[MACHINE_NAME] ([machine_name]\[user_name])
            Web Sites
                    Default Web Site
```

   Note: This activates CSP for the whole web site. To restrict the use of CSP to specific virtual sub-directories (such as /csp/) focus control on the appropriate subdirectory (under **Default Web Site**) before creating the mappings. Repeat the process for each virtual subdirectory from which CSP content is to be served.

3. In the middle panel, double-click the **Handler Mappings** icon.
4. In the right panel, select **Add Script Map**.
5. In the **Add Script Map** dialogue, enter:

   Request Path: *.csp

   Executable: C:\inetpub\CSPGateway\nph-CSPcgi.exe

   Name:CSPGateway_csp

6. Select **Request Restrictions**.

Clear: **Invoke handler only if request is mapped to**

Select **OK** to return to the **Add Script Map** dialogue.

Select **OK**.

7. At this point you may be prompted as follows: “Would you like to enable this ISAPI extension? If yes, we add your extension as an “Allowed” entry in the ISAPI and CGI Restrictions list. If the extension already exists we allow it.”

   Select **Yes**.

8. You can later find the list of allowed applications as follows:

   In the left panel, highlight:

   [MACHINE_NAME] ([machine_name]\[user_name])

   In the middle panel, double-click the **ISAPI and CGI Restrictions** icon.

   If the Gateway CGI components are not included in the list of allowed applications then add them (as you would have done for IIS 6):

   You can add text of your own choice in the **Description** field. For example:

   ```
   CSPGatewayManagement for nph-CSPcgiSys.exe
   ```

   ```
   CSPGatewayRunTime for nph-CSPcgi.exe
   ```

9. Repeat the above process: Use the **Add Script Map** dialogue to enter the following two mappings:

   Request Path: *.cls

   Executable: C:\inetpub\CSPGateway\nph-CSPcgi.exe

   Name: CSPGateway_cls

   Request Path: *.zen

   Executable: C:\inetpub\CSPGateway\nph-CSPcgi.exe

   Name: CSPGateway_zen

   Request Path: *.cxw

   Executable: C:\inetpub\CSPGateway\nph-CSPcgiSys.exe

   Name: CSPGatewayManagement

A.2.5.3 Operating and Managing the Gateway

This connectivity option depends on the CSP Gateway’s Network Service Daemon (NSD).

1. Start the CSP NSD as described in the section dedicated to this service.

To access the CSP Gateway’s Systems Management suite, point your browser at one of the following locations:

http://<ip_address>/csp/bin/Systems/Module.cxw

http://<ip_address>/csp-bin/nph-CSPcgiSys

The CSP engine is automatically invoked for requested files that contain a .csp, .cls, or .zen extension. For example:

http://<ip_address>/csp/samples/menu.csp

If you see an unauthorized user error message, refer to the section “CSP Gateway and Security”.

A.3 Alternative Options for IIS6 or Earlier

A.3.1 Using the ISAPI Filter (CSPmsf1.dll)

You can use this filter with all IIS connectivity options. It can be deployed to provide essential functionality in the following two areas.

A.3.1.1 Processing WebDAV Requests

The use of the filter is essential if CSP is used to implement WebDAV services.

Many Microsoft WebDAV clients include the `Translate: f` header in the HTTP request headers sent to the web server with each and every request. IIS, on detecting this header directive, attempts to process the request directly without forwarding it on to any further ISAPI extensions (such as the CSP Gateway) which might otherwise have been called in the absence of this header. This behavior effectively prohibits CSP from processing WebDAV requests.

The `Translate: f` header is essentially a way of avoiding the overloading of the GET method in the WebDAV protocol. HTTP GET usually means get (that is, run) the page; WebDAV clients expect this method to `get` the source to the page instead. IIS cannot possibly implement this latter functionality for CSP-based content because the physical content (or source code) is associated with the Caché server and not the web server. Therefore, requests from a Microsoft WebDAV client working to a CSP-based WebDAV server through IIS fail with HTTP `Forbidden` or `File doesn't exist` errors.

The filter works around this problem by examining the incoming request stream and translating `Translate: f` header directives to `Translate: g`. IIS then passes the request on to the CSP Gateway, if appropriate.

A.3.1.2 Processing Multiline HTTP Request Headers

IIS does not correctly process header directives that are split over multiple lines. In fact the whole HTTP header block can become badly corrupted.

Recent tests demonstrated that in some cases the affected header block can become corrupted to the extent that it is not possible to always work around the problem in the Gateway code (that is, after the corruption had occurred). See the example below.

The filter corrects this problem by removing carriage-return-linefeeds (CRLs) from individual header directives before IIS has a chance to parse the header block.

Example of the problem: Consider the following request header block.

```
POST /csp/xds/XDSRequest.csp HTTP/1.1
Accept: text/html, text/plain, text/xml, image/gif, image/jpeg, */*
Content-Length: 1787
certAlias: unknowncert
SOAPAction: \"" Content-Type: multipart/related; type="text/xml";" \
boundary=--boundary
421.41176470588235291359.470588235294118--
User-Agent: HttpClient/1.4.2
Mozilla/4.0
Host: localhost
Connection: keep-alive
```

Notice that the content boundary (part of the content-type directive) has been completely misplaced. It has been found that the nature of this corruption is not consistent. The servicing of the request can completely fail depending on the nature of the damage caused and the misparsing that occurs as a result.

A.3.1.3 Installing the Filter

The filter operates on raw request data and must therefore be installed globally for the whole web server:

1. Open the **Internet Services Manager**.
2. Navigate to **Web Sites**. Right-click this item and select **Properties**.
3. Select the **ISAPI Filters** tab.
4. Select **Add**.
5. Enter `CSP Gateway` for the **Filter** name.
6. Browse to CSPmsf1.dll for **Executable**.
7. Select **OK**.
8. Restart the **World Wide Web Publishing** service from the Windows **Services** control panel (not the Internet Information Services control panel). Alternatively, restart the computer.

A.3.2 Alternative Option 1: IIS and ISAPI Module with NSD (CSPcms.dll)

If you are using the ISAPI modules with the NSD with the IIS web server, follow the directions in this section.

In most cases, the all-inclusive ISAPI-based solution (the Recommended Option) is the option of choice, and is the implementation that gives the best performance. The ISAPI/NSD hybrid, described here, is useful for cases where it is necessary, for operational reasons, to manage the Gateway independently of the hosting web server; for example, if multiple instances of the web server are to share the same Gateway installation. In the Recommended Option, each instance of the core web server process binds to its own instance of the Gateway.

The Recommended Option provides better performance than the CGI/NSD hybrid described in Option 3. The higher latency that results from the need to start new processes to serve each and every request is avoided in this implementation.

A.3.2.1 Internet Information Services with ISAPI and NSD

If you are running any version of IIS using the ISAPI modules with the NSD, follow these directions:

Follow the instructions for the Recommended Option with the exception that CSP files should be associated with CSPcms.dll instead of CSPms.dll (steps 7 and 8) and nph-CSPcgiSys.exe instead of CSPmsSys.dll (step 9).

- Executable: *install-dir*\csp\bin\CSPcms.dll
- Extension: `csp`
- All Verbs: Select
- Script engine: Select
- Check that file exists: Clear

- Executable: *install-dir*\csp\bin\CSPcms.dll
- Extension: `cls`
- All Verbs: Select
- Script engine: Select
- Check that file exists: Clear

- Executable: *install-dir*\csp\bin\CSPcms.dll

- Extension: zen
- All Verbs: Select
- Script engine: Select
- Check that file exists: Clear

- Executable: *install-dir*\csp\bin\nph-CSPcgiSys.exe
- Extension: cxw
- All Verbs: Select
- Script engine: Select
- Check that file exists: Clear

Refer to the following section for further information relating to version 6 of IIS (shipped with Windows 2003).

A.3.2.2 Internet Information Services v6 with ISAPI and NSD

If you are running IIS 6, using the ISAPI modules with the NSD, follow the directions in the previous section and also follow these directions:

Follow the instructions for the Recommended Option with the exception that the following executables should be registered as allowed for the CSP Gateway instead of CSPms.dll and CSPmsSys.dll.

- CSPcms.dll
- nph-CSPcgi.exe
- nph-CSPcgiSys.exe

To Prohibit Access to CSP

Mark the following executables as prohibited:

- CSPcms.dll
- nph-CSPcgi.exe
- nph-CSPcgiSys.exe

To Prohibit Access to the CSP Gateway Systems Management Portal

Mark the following executables as prohibited:

- nph-CSPcgi.exe
- nph-CSPcgiSys.exe

To Prohibit Access to the CSP Gateway Runtime Module

Mark the following executable as prohibited: CSPcms.dll

A.3.2.3 Operating and Managing the Gateway with ISAPI and NSD

This connectivity option depends on the CSP Gateway Network Service Daemon (NSD).

1. Start the CSP NSD as described in "Operating the Network Service Daemon (NSD)".
2. Restart Apache after making changes to its configuration file (httpd.conf).

   The order in which you start Apache and the NSD is unimportant.

3. To access the CSP Gateway Systems Management Portal, point your browser at one of the following locations.

   Although CSP pages are served through the higher-performing module (mod_csp24.so), the CSP Web Gateway Management Page is accessed through the CGI module dedicated to this purpose (nph-CSPcgiSys).

   ```
   http://localhost:<port_no>/csp/bin/Systems/Module.cxw
   http://localhost:<port_no>/csp-bin/nph-CSPcgiSys
   ```

If you see an `Unauthorized User` error message, refer to the section on security considerations.

The CSP engine is automatically invoked for requested files that contain a .csp, .cls, or .zen extension. For example:

```
http://localhost:<port_no>/csp/samples/menu.csp
```

A.3.3 Alternative Option 2: IIS and CGI Modules with NSD (nph-CSPcgi.exe)

If you are using the CGI modules with the IIS web server with the NSD, follow the directions in this section:

In most cases, the all-inclusive ISAPI-based solution (Option 1) is the option of choice, and is the implementation that gives the best performance. The CGI/NSD hybrid is useful for cases where it is necessary, for operational reasons, to manage the Gateway independently of the hosting web server, for example, if multiple instances of the web server are to share the same Gateway installation. In Option 1, each instance of the core web server process binds to its own instance of the Gateway.

Another factor in choosing this approach might be that the in-house requirements of your web master (or ISP) dictate that all web server extensions are implemented using the CGI protocol.

A.3.3.1 Internet Information Services with CGI and NSD

If you are running any version of IIS using the CGI modules with the NSD, follow these directions:

Follow the instructions for Recommended Option with the exception that CSP files should be associated with nph-CSPcgi.exe instead of CSPms.dll (steps 7 and 8) and nph-CSPcgiSys.exe instead of CSPmsSys.dll (step 9).

- Executable: *install-dir*\csp\bin\nph-CSPcgi.exe
- Extension: `csp`
- All Verbs: Select
- Script engine: Select
- Check that file exists: Clear

- Executable: *install-dir*\csp\bin\nph-CSPcgi.exe
- Extension: `cls`
- All Verbs: Select
- Script engine: Select
- Check that file exists: Clear

- Executable: *install-dir*\csp\bin\nph-CSPcgi.exe
- Extension: `zen`
- All Verbs: Select
- Script engine: Select
- Check that file exists: Clear

- Executable: *install-dir*\csp\bin\nph-CSPcgiSys.exe
- Extension: cxw
- All Verbs: Select
- Script engine: Select
- Check that file exists: Clear

Refer to the following section for further information relating to version 6 of IIS (shipped with Windows 2003).

A.3.3.2 Internet Information Services v6 with CGI and NSD

If you are running IIS 6, follow the directions in the previous section for all versions of IIS and also follow the directions in this section.

Follow the instructions for Recommended Option IIS 6, except register the following executables as allowed for the CSP Gateway instead of CSPms.dll and CSPmsSys.dll:

- nph-CSPcgi.exe
- nph-CSPcgiSys.exe

To Prohibit Access to CSP

Mark the following executables as prohibited:

- nph-CSPcgi.exe
- nph-CSPcgiSys.exe

To Prohibit Access to the CSP Gateway Systems Management Module

Mark the following executable as prohibited: nph-CSPcgiSys.exe.

To Prohibit Access to the CSP Gateway Runtime Module

Mark the following executable as prohibited: nph-CSPcgi.exe.

A.3.3.3 Operating and Managing the Gateway with CGI

This connectivity option depends on the CSP Gateway's Network Service Daemon (NSD).

1. Start the CSP NSD as described in "Operating the Network Service Daemon (NSD)".
2. Restart Apache after making changes to its configuration (httpd.conf).

   The order in which Apache and the NSD are started is unimportant.
3. To access the CSP Web Gateway Management page, point your browser at one of the following locations. Although CSP pages are served through the higher-performing module (mod_csp24.so), the CSP Web Gateway Management page is accessed through the CGI module dedicated to this purpose (nph-CSPcgiSys).

   ```
   http://localhost:<port_no>/csp/bin/Systems/Module.cxw
   http://localhost:<port_no>/csp-bin/nph-CSPcgiSys
   ```

If you see an `Unauthorized User` error message, refer to the section on security considerations.

The CSP engine is automatically invoked for requested files that contain a .csp, .cls, or .zen extension, such as `http://localhost:<port_no>/csp/samples/menu.csp`

A.4 Alternative Options for Windows Apache

This section contains information for installations with Apache web servers on Microsoft Windows. Read the sections that apply to your installation.

Note: *If you are using Apache 2.2 or earlier*: If you use the directions in this section, you add text to the end of the httpd.conf file. The text added specifying access control needs to be different for Apache 2.4 or Apache 2.2 (and earlier). If you are using Apache 2.4, use the text shown in this section. If you are using Apache 2.2, replace the following phrases in the httpd.conf sections. Replace `Require all denied`, with the line `Deny from all`. Replace `Require all granted` with the two lines `Order allow,deny` and `Allow from all`. If you require more information, see http://httpd.apache.org/docs/2.4/upgrading.html

A.4.1 Install Locations

The following modules are installed:

- CSPcgi.exe (Runtime module)
- nph-CSPcgi.exe (Copy of CSPcgi)
- CSPcgiSys.exe (Systems-Management module)
- nph-CSPcgiSys.exe (Copy of CSPcgiSys)

Note: There are separate binaries for each version of the Apache server as shown below.

Apache Version 2.4.x

- mod_csp24.dll (Apache built-in momod_dule as a DLL, if supplied)
- CSPa24.dll (Runtime module, if supplied)
- CSPa24Sys.dll (Gateway Systems Management module, if supplied)

Apache Version 2.2.x

- mod_csp22.dll (Apache built-in momod_dule as a DLL, if supplied)
- CSPa22.dll (Runtime module, if supplied)
- CSPa22Sys.dll (Gateway Systems Management module, if supplied)

Apache Version 2.0.x:

- mod_csp2.dll (Apache built-in module as a DLL, if supplied)
- CSPa2.dll (Runtime module, if supplied)
- CSPa2Sys.dll (Gateway Systems Management module, if supplied)

The default location for these binaries is:

C:\Program Files\Apache Group\Apache\CSPGateway\bin

The original location (*install-dir*\csp\bin) is used to hold the Gateway components required for serving the Management Portal for the specific instance of Caché.

The configuration file (CSP.INI) and Event Log (CSP.LOG) are written in this directory for non NSD-based connectivity options.

The modules with Sys appended are special modules for accessing the CSP Web Gateway Management page. The runtime modules (that is, those without Sys) have no access to the systems management forms.

A.4.2 Alternative Option 1: Apache and CGI Modules with NSD (nph-CSPcgi.exe)

Configure the web server such that it recognizes CSP requests (files of type .csp, .cls, and .zen) and passes them to the CSP gateway for processing.

The web server configuration file (httpd.conf) is in the following directory:

C:\Program Files\Apache Group\Apache\conf

Add the following section to the end of httpd.conf:

```
<LocationMatch "/*\.([Cc][Ss][Pp]|[Cc][Ll][Ss]|[Zz][En][Nn])$">
    AllowOverride None
    Options FollowSymLinks ExecCGI
    Require all granted
</LocationMatch>
ScriptAliasMatch /*\.([Cc][Ss][Pp]|[Cc][Ll][Ss])$ "c:/cache-install-dir/csp/bin/nph-CSPcgi.exe"
Alias /csp/ c:/cache-install-dir/csp/
<Directory "c:/cache-install-dir/csp">
    AllowOverride None
    Options MultiViews FollowSymLinks ExecCGI
    Require all granted
    <FilesMatch "\.(log|ini|pid|exe)$">
        Require all denied
    </FilesMatch>
</Directory>
ScriptAlias /csp-bin/ "c:/cache-install-dir/csp/bin/"
ScriptAliasMatch /csp/bin/Systems/Module.cxw "c:/cache-install-dir/csp/bin/nph-CSPcgiSys.exe"
ScriptAliasMatch /csp/bin/RunTime/Module.cxw "c:/cache-install-dir/csp/bin/nph-CSPcgi.exe"
<Directory "c:/cache-install-dir/csp/bin/">
    AllowOverride None
    Options None
    Require all granted
    <FilesMatch "\.(exe)$">
        Allow from all
    </FilesMatch>
</Directory>
```

The above configuration block relies on the Regular Expressions (`regex`) processor being available to the Apache environment. Sometimes this is not the case (particularly with Windows 2000 systems) and CSP files are consequently not served (`File not found` errors are returned). To remedy this situation, associate the (virtual) root location of your CSP applications with the CGI module instead of making the association through the CSP file extensions. For example, your CSP applications are in /csp. To associate the CSP CGI module with files under /csp, replace the following configuration block:

```
<LocationMatch "/*\.([Cc][Ss][Pp]|[Cc][Ll][Ss]|[Zz][En][Nn])$">
    AllowOverride None
    Options FollowSymLinks ExecCGI
    Require all granted
</LocationMatch>
ScriptAliasMatch /*\.([Cc][Ss][Pp]|[Cc][Ll][Ss])$
      "c:/cache-install-dir/csp/bin/nph-CSPcgi.exe"
```

with

```
<Location "/csp">
    AllowOverride None
    Options FollowSymLinks ExecCGI
    Require all granted
</Location>
ScriptAlias /csp "c:/cache-install-dir/csp/bin/nph-CSPcgi.exe"
```

These directives work for URLs of the form:

```
http://localhost:<port_no>/csp/*.csp
```

Duplicate this configuration block for other root locations. For example, repeat the process for /myapps for URLs of the form:

```
http://localhost:<port_no>/myapps/*.csp
```

Another approach to avoiding the `regex` issue is to use an `Action` directive in conjunction with a CSP MIME type. However, note that `Action` is a content filtering technique and, as such, requires that your CSP files are physically present on the web server host even if the Caché server is installed on a separate computer.

To use this approach:

1. Add a new MIME type to the end of the Apache mime.types file and associate it with the file types representing CSP content, .csp, .cls, and .zen. The mime.types file are in the same directory as the httpd.conf file:

   ```
   text/csp                csp cls
   ```

2. Add the `Action` directive to the end of the CGI configuration block in httpd.conf such that it reads:

   ```
   Alias /csp/ c:/cache-install-dir/csp/
   <Directory "c:/cache-install-dir/csp">
       AllowOverride None
       Options MultiViews FollowSymLinks ExecCGI
       Require all granted
   <Files CSPnsd.exe>
          Require all denied
       </Files>
   <Files CSP.ini>
          Require all denied
       </Files>
   <Files CSP.log>
          Require all denied
       </Files>
   <Files CSPnsd.ini>
          Require all denied
       </Files>
   <Files CSPnsd.pid>
          Require all denied
       </Files>
       <FilesMatch "\.(log|ini|pid|exe)$">
          Require all denied
       </FilesMatch>
   </Directory>
   ScriptAlias /csp-bin/ "c:/cache-install-dir/csp/bin/"
   <Directory "c:/cache-install-dir/csp/bin/">
       AllowOverride None
       Options None
       Require all granted
   </Directory>
   Action text/csp “/csp-bin/nph-CSPcgi.exe”
   ```

   Finally, note that because CGI is an open standard, the CSP CGI modules work with any web server.

3. Restart Apache after making changes to httpd.conf.

A.4.2.1 Registering Additional File Types with CSP

Apache API modules always recognize the following reserved file extensions:

```
.csp .cls .zen .cxw
```

You may have other files that you want to send to CSP for processing. For example, if you need to serve other static files through the CSP Gateway or need to access the Management Portal through this web server, add mappings for file types .jpg, .gif, .png, .css, and .js.

You can configure Apache to recognize what files to pass on to CSP in any of the following ways:

- By `CSP` location directive
- By file extension – `CSPFileTypes` directive
- By MIME type

By CSPlocation directive

Use the CSP directive to request that all files within a certain location be processed by CSP. The following requests that all files and directories under the /csp path be processed by CSP.

```
<Location /csp>
   CSP On
   SetHandler csp-handler-sa
</Location>
```

For example, all the following would be sent to CSP for processing:

```
/csp/
csp/samples/menu.csp
csp/sys/
```

By file extension – CSPFileTypes directive

The `CSPFileTypes` directive works for requests for files that have extensions (such /csp/menu.csp). It does not work for requests for files that do not have file extensions (such as/csp/menu).

This parameter is processed by the Gateway's Apache modules and can be globally defined at the server definition level (in httpd.conf) or restricted within the definition for a location or directory block.

By file type: The following directive requests that files of type xxx and yyy be processed by CSP.

```
CSPFileTypes xxx .yyy
```

By location: The following requests that files of type xxx and yyy be processed by CSP but only for locations under /csp (including subdirectories, such as /csp/samples and so on).

```
<Location /csp/>
   CSPFileTypes xxx yyy
</Location>
```

Using the wildcard character, the following requests that all files under path /csp (and /csp/samples and so on) be processed by CSP.

```
<Location /csp/>
   CSPFileTypes *
</Location>
```

By MIME type

In addition to recognizing the file extensions listed above, CSP can also recognize files for the following MIME types:

```
application/x-csp
```

and

```
text/csp
```

For example, to add the file extension xxx to the list of files processed by CSP, use:

```
LoadModule csp_module_sa /cache-install-dir/csp/bin/CSPa22.dll
AddType application/x-csp csp cls zen xxx
```

One of the problems with using MIME types to associate types of file with CSP is that Apache checks to ensure that the path to the resource (that is, the hosting directory) physically exists, and returns a `file not found` error if it does not. It does not, however, check to ensure that the file requested physically exists – which is appropriate for resources served by CSP since they are served by Caché and are virtual as far as the web server is concerned. The "By MIME type" approach is therefore only suitable for cases where the application's path structure can be replicated on the web server.

A.4.2.2 Operating and Managing the Gateway with Apache NSD

This connectivity option depends on the CSP Gateway's Network Service Daemon (NSD).

1. Start the CSP NSD as described in "Operating the Network Service Daemon".
2. Restart Apache after making changes to its configuration (httpd.conf).

   The order in which Apache and the NSD are started is unimportant.
3. To access the CSP Web Gateway Management page, point your browser at one of the following locations:

```
http://localhost:<port_no>/csp/bin/Systems/Module.cxw
http://localhost:<port_no>/csp-bin/nph-CSPcgiSys
```

If you see an `Unauthorized User` error message, refer to the section on security considerations.

The CSP engine is automatically invoked for requested files that contain a .csp, .cls, or .zen extension. For example:

```
http://localhost:<port_no>/csp/samples/menu.csp
```

A.4.3 Alternative Option 2: Apache API Module with NSD (mod_csp.dll)

Note: This connectivity option is not used as often as the stand-alone API modules described in Option 1, however, it can be used if you need to use the NSD. The CSP module, built as a DLL (`mod_csp24.dll` – for Apache 2.4), performs better than the CGI-based solution (Option 2). The module is usually named `mod_csp2.dll` for Apache v2.0.x, and `mod_csp22.dll` for Apache v2.2.x.

1. Edit the Apache configuration file httpd.conf. For the standard Apache distribution this file is in:

   C:\Program Files>\Apache Group\Apache\conf

   To invoke CSP for files with the .csp, .cls, and .zen extensions, add the following section to the end of httpd.conf. For Apache v2.4.x, specify `mod_csp24.dll`. For Apache v2.2.x, specify `mod_csp22.dll`). For Apache v2.0.x, specify `mod_csp2.dll`.

```
LoadModule csp_module c:/cache-install-dir/csp/bin/mod_csp24.dll
<LocationMatch "/*\.([Cc][Ss][Pp]|[Cc][Ll][Ss]|[Zz][En][Nn])$">
                SetHandler csp-handler
</LocationMatch>
Alias /csp/ /cache-install-dir/csp/
<Directory "c:/cache-install-dir/csp">
        AllowOverride None
        Options MultiViews FollowSymLinks ExecCGI
        Require all granted
        <FilesMatch "\.(log|ini|pid|exe)$">
                Require all denied
        </FilesMatch>
        <Files CSPnsd>
                Require all denied
        </Files>
</Directory>
ScriptAlias /csp-bin/ "c:/cache-install-dir/csp/bin/"
ScriptAliasMatch /csp/bin/Systems/Module.cxw \
         "c:/cache-install-dir/csp/bin/nph-CSPcgiSys.exe"
ScriptAliasMatch /csp/bin/RunTime/Module.cxw \
         "c:/cache-install-dir/csp/bin/nph-CSPcgi.exe"
<Directory "c:/cache-install-dir/csp/bin/">
        AllowOverride None
        Options None
        Require all granted
</Directory>
```

2. Optional step, if required: The above configuration block expects that the Regular Expressions (`regex`) processor is available to the Apache environment. If this is not the case (particularly with Windows 2000 systems), CSP files are not served (`File not found` errors are returned). To remedy this situation, replace the following configuration block:

```
<LocationMatch "/*\.([Cc][Ss][Pp]|[Cc][Ll][Ss]|[Zz][En][Nn])$">
                SetHandler csp-handler
</LocationMatch>
```

with:

```
CSPFiletypes csp cls
```

3. Restart Apache after making changes to httpd.conf.

A.4.3.1 Registering additional file types with CSP

To configure additional file types to be processed by CSP, include the new file extension(s) in the list of usual file extensions (csp,. cls, zen) to be processed by the CGI module. For example, add them to the following line:

```
ScriptAliasMatch /*\.([Cc][Ss][Pp]|[Cc][Ll][Ss]|[Zz][En][Nn])$  "/cache-install-dir/csp/bin/nph-CSPcgi"
```

If you need to serve other static files through the CSP Gateway or need to access the Management Portal through this web server, add mappings for file types .jpg, .gif, .png, .css, and .js.

The following directive can be used to map requests for all files to CSP for a given path.

```
ScriptAliasMatch ^/csp/*/.* "/cache-install-dir/csp/bin/nph-CSPcgi"
```

Therefore, a basic configuration block for mapping requests for all files in the /csp path to CSP would be:

```
ScriptAliasMatch ^/csp/*/.* "/cache-install-dir/csp/bin/nph-CSPcgi"
Directory "/cache-install-dir/csp/bin/">
   AllowOverride None
   Options None
   Require all granted
</Directory>
```

A.4.3.2 Operating and Managing the Gateway with Apache API and NSD

This connectivity option depends on the CSP Gateway's Network Service Daemon (NSD).

1. Start the CSP NSD as described in "Operating the Network Service Daemon".
2. Restart Apache after making changes to its configuration (httpd.conf).

   The order in which Apache and the NSD are started is unimportant.
3. To access the CSP Web Gateway Management page, point your browser at one of the following locations.

   Although CSP pages are served through the higher-performing module (mod_csp24.dll), the CSP Web Gateway Management page is accessed through the CGI module dedicated to this purpose (nph-CSPcgiSys.exe).

   ```
   http://localhost:<port_no>/csp/bin/Systems/Module.cxw
   http://localhost:<port_no>/csp-bin/nph-CSPcgiSys.exe
   ```

If you see an `Unauthorized User` error message, refer to the section on security considerations.

The CSP engine is automatically invoked for requested files that contain a .csp, .cls, or .zen extension. For example:

```
http://localhost:<port_no>/csp/samples/menu.csp
```

A.4.4 Alternative Option 3: Apache and ISAPI Modules (CSPms.dll)

Note: This connectivity option is superseded by the stand-alone API modules described in the Recommended Option and should not be used. It is documented here as a reference for legacy systems that have used it in the past.

The Apache Group provides a module that attempts to emulate Microsoft's ISAPI interface. If Apache is configured to use this module then ISAPI extensions may be run. However, there are significant differences between the Apache Group ISAPI interface and Microsoft's original. The most troublesome feature of the Apache ISAPI module is that it unloads its ISAPI extensions (DLLs) after servicing each and every request. This behavior is unacceptable for CSP because the CSP Gateway relies on its ISAPI DLLs remaining in memory in order for it to manage a persistent pool of connections to Caché.

The modified ISAPI module supplied with CSP allows the CSP Gateway's ISAPI extensions to remain loaded between requests. The modifications only affect the Gateway's ISAPI DLLs; all other ISAPI DLLs are subject to the original Apache Group's functionality.

A.4.4.1 Rebuilding the Apache Executable

1. Upgrade the Apache ISAPI module (mod_isapi.c)

   Overwrite the Apache Group's ISAPI module with the version contained in the CSP distribution:

   C:\Program Files\Apache Group\Apache\src\os\win32\mod_isapi.c

2. Rebuild the Apache executable

   In order to perform this step you need version 5.0 (or later) of the Microsoft C Compiler (Microsoft Visual C++).

   Change to the following directory:

   C:\Program Files\Apache Group\Apache\src\os\win32\mod_isapi.c

   Build Apache with:

   ```
   nmake /f Makefile.nt installr INSTDIR=d:\progra~1\apache~1\apache
   ```

   You can safely ignore the many warning messages that the build process displays.

3. Runtime configuration

   Edit the Apache configuration file httpd.conf. For the standard Apache distribution this file is in:

   C:\Program Files\Apache Group\Apache\conf

   Assuming that you wish to invoke the CSP engine for requested files that contain a .csp, .cls, or .zen extension, add the following section to the end of httpd.conf:

   ```
   AddHandler isapi-isa dll
   AddHandler isapi-isa csp
   AddHandler isapi-isa cls
   AddHandler isapi-isa zen
   AddHandler isapi-isa cxw
   Alias /csp/ /cache-install-dir/csp/
   <Directory "c:/cache-install-dir/csp">
           AllowOverride None
           Options MultiViews
   FollowSymLinks ExecCGI
           Require all granted
           <FilesMatch ".(log|ini|pid|exe)$">
                   Require all denied
           </FilesMatch>
   </Directory>
   Alias /csp/bin/Systems/Module.cxw
   /csp/bin/CSPmsSys.dll
   ```

4. Restart Apache after making changes to httpd.conf.

A.4.4.2 Operating and Managing the Gateway with Apache and ISAPI

To access the CSP Web Gateway Management page, point your browser at one of the following locations:

```
http://localhost:<port_no>/csp/bin/Systems/Module.cxw
http://localhost:<port_no>/csp/bin/CSPmsSys.dll
```

If you see an `Unauthorized User` error message, refer to the section on security considerations.

The CSP engine is automatically invoked for requested files that contain a .csp, .cls, or .zen extension. For example:

```
http://localhost:<port_no>/csp/samples/menu.csp
```

A.4.5 Locked-down Apache Environments for Microsoft Windows

Occasionally Apache is locked-down such that you cannot easily configure the server to access files outside the Apache file system.

For configurations locked down in this way, the CSP Gateway configurations discussed in previous sections result in `HTTP 403 Forbidden` error codes being returned when you try to access CSP resources. To work with these secure configurations is to copy the file system under:

cache-install-dir\csp\

to a location under the Apache root:

C:\Program Files\Apache Group\Apache\

Specify appropriate changes to the paths specified in the Apache configuration.

An alternative approach is to configure the CSP Gateway to work within the pre-configured directories provided by Apache.

1. Copy CGI modules to: C:\Program Files\Apache Group\Apache\cgi-bin\ as follows:

   ```
   copy c:\cache-install-dir\csp\bin\*cgi*.exe C:\Program Files\Apache Group\Apache\cgi-bin\
   ```

2. Copy API modules to C:\Program Files\Apache Group\Apache\modules:

   ```
   copy c:\cache-install-dir\csp\bin\*.dll C:\Program Files\Apache Group\Apache\modules\
   ```

3. Copy static files (and their subdirectories) to locations under C:\Program Files\Apache Group\Apache\htdocs\csp\samples.

   ```
   copy c:\cache-install-dir\csp\samples\*.* \
   C:\Program Files\Apache Group\Apache\htdocs\csp\samples\
   copy c:\cache-install-dir\csp\broker\*.* \
   C:\Program Files\Apache Group\Apache\htdocs\csp\broker\
   copy c:\cache-install-dir\csp\sys\*.* \
   C:\Program Files\Apache Group\Apache\htdocs\csp\sys\
   ```

4. Install the NSD component (if required) in C:\Program Files\Apache Group\Apache\nsd.

Using the pre-configured directories in Apache simplifies the CSP Gateway configuration in httpd.conf. Modified configuration blocks are shown below.

A.4.5.1 Configuration for Recommended Option: Apache API Modules (CSPa.dll)

```
LoadModule csp_module_sa
    C:/Program Files/Apache Group/Apache/modules/CSPa24.dll
<Location "/csp/bin/Systems/">
    SetHandler csp-handler-sa
</Location>
<Location "/csp/bin/RunTime/">
    SetHandler csp-handler-sa
</Location>
CSPFiletypes csp cls zen cxw
```

A.4.5.2 Configuration for Atypical Option 2: CGI Modules with NSD (nph-CSPcgi.exe)

```
<LocationMatch "/*\.([Cc][Ss][Pp]|[Cc][Ll][Ss]|[Zz][En][Nn])$">
AllowOverride None
Options FollowSymLinks ExecCGI
Require all granted
</LocationMatch>
ScriptAliasMatch /csp/bin/Systems/Module.cxw "C:/Program Files/Apache
Group/Apache/cgi-bin/nph-CSPcgiSys.exe"
ScriptAliasMatch /csp/bin/RunTime/Module.cxw "C:/Program Files/Apache Group/Apache/cgi-bin/nph-CSPcgi.exe"
ScriptAliasMatch /*\.([Cc][Ss][Pp]|[Cc][Ll][Ss])$ "C:/Program Files/Apache
Group/Apache/cgi-bin/nph-CSPcgi.exe"
```

A.4.5.3 Configuration for Atypical Option 3: Apache API Module with NSD (mod_csp24.dll)

```
LoadModule csp_module \
         C:/<cache-install-dir>/Apache Group/Apache/modules/mod_csp24.dll
<LocationMatch "/*\.([Cc][Ss][Pp]|[Cc][Ll][Ss]|[Zz][En][Nn])$">
SetHandler csp-handler
</LocationMatch>
ScriptAliasMatch /csp/bin/Systems/Module.cxw "C:/Program Files/Apache
Group/Apache/cgi-bin/nph-CSPcgiSys.exe"
ScriptAliasMatch /csp/bin/RunTime/Module.cxw "C:/Program Files/Apache Group/Apache/cgi-bin/nph-CSPcgi.exe"
```

A.4.5.4 Configuration for Atypical Option 4: ISAPI Modules (CSPms.dll)

Legacy only

```
AddHandler isapi-isa dll
AddHandler isapi-isa csp
AddHandler isapi-isa cls
AddHandler isapi-isa zen
AddHandler isapi-isa cxw
Alias /csp/bin/Systems/Module.cxw /csp/bin/CSPmsSys.dll
```

B

Alternative Configurations for UNIX®, Linux, and Mac OS X

This section describes how to configure atypical options for UNIX, Linux, and Mac OS.

B.1 Using the NSD on UNIX®, Linux, Mac OS X

B.1.1 When to Use the NSD

Several of the options described in this book use the NSD. There are two situations in which you might choose to use the NSD to separate the CSP Gateway from the web server so that you can manage the CSP Gateway independently of the Web Server. These are:

- If your web server distributes its load over multiple server processes, an instance of the CSP Gateway is then attached to each web server process.
- If you have a very large web server installation for which CSP is only a small part; for example, a web server that serves php, static content, .NET, and .ASP applications, as well as CSP applications.

B.1.2 NSD Module Install Locations

The NSD Module, if required, is CSPnsd.

The default location for this module is:

```
/opt/cspgateway/bin
```

The NSD should be run from within its home directory (above). The configuration file (CSP.INI) and Event Log (CSP.LOG) are written in this directory for NSD-based connectivity options.

B.1.3 Operating the NSD

To run the NSD:

1. Change to the following directory:

   /opt/cspgateway/csp

2. Enter the following command to start the NSD:

   ```
   ./CSPnsd
   ```

Before retiring to the background, the NSD displays a banner indicating its running configuration. It shows the TCP port number dedicated to this service, which is, by default, port number 7038.

You can suppress all startup messages for this command using the `-s` qualifier. For example, to start the NSD from a script invoked at system boot, use:

```
/opt/cspgateway/csp/CSPnsd –s
```

Other common startup options:

- Display help information.

  ```
  ./CSPnsd -h
  ```

- Pause the operation of the NSD. This command sends a stop signal (SIGSTOP) to the NSD process.

  ```
  ./CSPnsd -pause
  ```

- Continue the operation of the NSD (after a pause). This command sends a continue signal (SIGCONT) to the NSD process.

  ```
  ./CSPnsd -cont
  ```

- Give permission to others to run the NSD. Administrators of the NSD (CSPnsd) component can give permission to a group or others to start/stop the NSD using `CSPnsd –m=`*s* where *s* is a startup option.

  s can be one of

  – `u` for the current user (default)

  – `g` for the current group

  – `o` for others

  – `a` for everyone (`m=ugo`)

  Example: `CPSnsd -m=ug` gives permissions to the group (the Administrator group) to run the NSD. This command gives the CPSnsd.pid permissions of: `-rw-rw---`

  When the command to stop the CSPnsd is issued, it tries to signal the CSPnsd parent process to shut down as before. If this is not possible because the service was started by a different user, a flag is written to the CSPnsd.ini file and the service gracefully closes itself down when it acknowledges this flag. This process takes up to 20 seconds to complete.

To close down the NSD, enter:

```
./CSPnsd -stop
```

Alternatively:

```
kill –TERM `cat /opt/cspgateway/csp/CSPnsd.pid`
```

These commands close down the NSD in an orderly manner – it gracefully terminates all open connections to Caché and releases all its system resources before terminating. Do not use the **kill –9** command to terminate the NSD.

All errors are reported in the Event Log (CSP.log). This file is created and maintained in the NSD home directory (such as /opt/cspgateway/csp). The configuration file CSP.ini also resides in this directory.

B.1.3.1 Starting the NSD on Alternative TCP Port

By default, the NSD listens for incoming requests on TCP port 7038. You can override this by starting the service as follows, where *port_no* is the TCP port number of your choice.

```
./CSPnsd [port_no]
```

Or:

```
./CSPnsd -p=[port_no]
```

On startup, the NSD creates the following file:

/opt/cspgateway/csp/CSPnsd.ini

Typically, this file contains the following lines:

```
[SYSTEM]
Ip_Address=127.0.0.1
TCP_Port=7038
```

In this context, the clients are the CSP module contained within, or dynamically linked to, the web server and/or the CSP CGI modules invoked by the server. It is, therefore, essential that this file is not deleted or moved. It is also important that the web server processes can read this file. Set the privileges accordingly, bearing in mind the UNIX® username under which your web server is operating. The NSD clients attempt to find this file in the following locations:

/cache-install-dir/csp

/opt/cspgateway/csp

/etc

If the NSD is operating in a different directory, you have to move the CSPnsd.ini file to one of the locations listed.

Storing the NSD port number in the CSPnsd.ini file is inappropriate for situations in which multiple instances of the NSD are running. For Apache servers there is a much better mechanism for communicating the TCP port number of the NSD to its clients. Set the following environment variables in the Apache configuration to indicate the address and port of the target NSD installation.

CSP_NSD_NAME — This is the IP address of the NSD. Only use this parameter if the NSD is operating on a remote computer.

CSP_NSD_PORT — This is the TCP port of the NSD.

The values specified in these environment variables take precedence over any values found in the CSPnsd.ini file.

Example 1:

To distribute the load for two Apache virtual hosts (123.123.1.1 and 123.123.1.2) between two independent NSD installations (listening on TCP port 7038 and 7039), add the following directives to the Apache configuration (httpd.conf):

```
<VirtualHost 123.123.1.1>
ServerName 123.123.1.1
SetEnv CSP_NSD_PORT 7038
</VirtualHost>
<VirtualHost 123.123.1.2>
ServerName 123.123.1.2
SetEnv CSP_NSD_PORT 7039
</VirtualHost>
```

Example 2:

To distribute the load for two CSP applications (/csp1 and /csp2) between two independent NSD installations (listening on TCP port 7038 and 7039), add the following directives to the Apache configuration (httpd.conf):

```
<Location /csp1>
SetEnv CSP_NSD_PORT 7038
</Location>
<Location /csp2>
SetEnv CSP_NSD_PORT 7039
</Location>
```

Restart Apache after making changes to its configuration.

In cases where multiple instances of the NSD are running, it is recommended that the separate instances be installed in separate directories, each maintaining its own copy of the configuration and log files (CSP.ini and CSP.log). The CSP Web Gateway Management page for each instance can easily be accessed by using the NSD's internal HTTP server. For example:

`http://localhost:7038/csp/bin/Systems/Module.cxw`

`http://localhost:7039/csp/bin/Systems/Module.cxw`

Spreading the Load over Multiple NSD Processes

By default, the NSD operates in a two-process mode of operation (one parent and one child worker).

However, there are limits to the number of threads that a single UNIX® process can start. If the concurrent load of the CSP application is resulting in requests queuing for available threads, consider raising the number of processes used by the NSD.

```
./CSPnsd –c=[no_processes]
```

- where no_processes is the number of child (or worker) processes to start.

It should be noted that there are even advantages in setting the number of child processes to one.

```
./CSPnsd –c=1
```

Under these circumstances, the NSD actually start two processes: a parent and one child worker process. The presence of the parent processes when using the '–c' directive improves the resilience of the NSD because if a fault develops in one of the worker processes the parent can replace the process. For the single, multithreaded architecture, the NSD cannot always recover from serious internal error conditions.

State-aware connectivity (preserve mode 1) should not be used in cases where the number of worker processes exceeds one.

Granting Administrator Rights to the NSD

Administrators of the NSD (CSPnsd) component can have some control over the user (or group) permitted to start/stop this service.

In the default scenario, the CSPnsd master process ID (PID) file (CSPnsd) is created such that only the user who started the service can subsequently close it down.

Administrators can now choose, for example, to allow all users belonging to the current UNIX group to manage the service. This is the group to which the administrating user belongs.

```
NSD start-up option: [-m=s]
    Define the user(s) permitted to manage this service
        where 's' is:
            'u' for the current user (the default),
            'g' for the current group,
            'o' for others,
            'a' for everyone (m=ugo),
```

Example:

```
./CSPnsd -m=ug
```

This allows the current user and all others in the current user's group to manage the NSD.

When the command to stop the NSD is issued, it first tries to signal the CSPnsd parent process to shut down as before. If this is not possible due to the service having been started by a different user, a flag is written to the CSPnsd.ini file and the service gracefully closes itself down when it acknowledges this flag. This process takes up to 20 seconds to complete.

B.2 Atypical Options for Apache for UNIX®, Linux, Mac OS

This section contains the following subsections. Read the first section for all atypical options. Then follow the directions in the option that applies to your installation.

1. Installing with Apache Servers on UNIX®, Linux, and Mac OS (All Alternative Options)
2. Alternative Option 1: Apache API Module with NSD (mod_csp24.so)
3. Alternative Option 2: CGI Modules with NSD (nph-CSPcgi)
4. Alternative Option 3: Built-in Apache API Module with NSD (mod_csp.c)

B.2.1 Install Locations Apache on UNIX®, Linux, Mac OS (All Alternative Options)

This section describes directory locations for CSP Gateway files and CSP static files.

1. The NSD module is:

   ```
   CSPnsd
   ```

   The default location of this module is:

   ```
   /opt/cspgateway/csp
   ```

   The NSD should be run from within its home directory /opt/cspgateway/csp. The configuration file, CSP.INI, and the event log, CSP.LOG, are written in this directory.

   In order to avoid disrupting existing Gateway installations on upgrading Caché, the installation places the following modules in the common location /usr/cspgateway/nsd. This location is not related to a particular Caché instance.

2. CGI and other dynamically-linked modules:

   - CSPcgi (Runtime module)
   - nph-CSPcgi (Copy of CSPcgi)
   - CSPcgiSys (Systems-Management module)
   - nph-CSPcgiSys (Copy of CSPcgiSys)
   - mod_csp22.so (Apache Version 2.4.x — Apache module as a DSO, if supplied)
   - mod_csp22.so (Apache Version 2.2.x — Apache module as a DSO, if supplied)
   - mod_csp2.so (Apache Version 2.0.x — Apache module as a DSO, if supplied)

   In order to avoid disrupting existing Gateway installations on upgrading Caché, the installation procedures place these modules in the following common location. This location is not related to a particular Caché instance.

   ```
   /usr/cspgateway/bin
   ```

The original location (/cache-install-dir/csp/bin) is used to hold the Gateway components required for serving the Management Portal for the specific instance of Caché.

The modules with Sys appended access the CSP Web Gateway Management page. The runtime modules (that is, those without Sys) have no access to the CSP Web Gateway Management pages.

3. The default location for the HyperEvents components:

   - CSPBroker.js
   - CSPxmlhttp.js

   and miscellaneous static resources (such as image files) are required by the CSP Samples and the Management Portal is:

   cache-install-dir\csp\broker

Requirements for using Apache API Modules (Recommended Option and Alternative Option 1)

Before following instructions for either the recommended option ("Recommended Option: NSAPI Modules (CSPn3.so)") or atypical option 1 ("Alternative Option 1: Apache API Module with NSD (mod_csp24.so)", check that your build of Apache includes the built-in module for managing shared objects (mod_so). To perform this check, run the following command which lists the modules currently available within Apache:

```
httpd -l
```

The shared object module (mod_so) should appear in the list of modules displayed. The following shows a typical module listing (with mod_so included):

```
Compiled in modules:
  core.c
  mod_access.c
  mod_auth.c
  mod_include.c
  mod_log_config.c
  mod_env.c
  mod_setenvif.c
  prefork.c
  http_core.c
  mod_mime.c
  mod_status.c
  mod_autoindex.c
  mod_asis.c
  mod_cgi.c
  mod_negotiation.c
  mod_dir.c
  mod_imap.c
  mod_actions.c
  mod_userdir.c
  mod_alias.c
  mod_so.c
```

If mod_so is not included in the list for your Apache installation, refer to your Apache documentation and follow the procedure for rebuilding Apache to include this module.

B.2.2 Atypical Option 1: Apache API Module with NSD (mod_csp24.so)

If the CSP module is supplied with your distribution as a pre-built shared object (mod_csp22.so or mod_csp2.so), then proceed to the section on configuration. To build the shared object from the supplied source file mod_csp.c choose Method 1 or Method 2 below. Method 1 is preferred.

Use module mod_csp24.so for Apache Version 2.4.x, mod_csp22.so for Apache Version 2.2.x, and mod_csp2.so for Apache Version 2.0.x.

Be sure to read the following instructions regarding the creation of shared objects in conjunction with the specific documentation contained within your Apache distribution. Note that the instructions given here assume that the root directory for the Apache installation is apache. In practice, this directory name usually has the Apache version number appended to it.

The module source file supplied (mod_csp.c) is fully compliant with both Apache v2.2.x and Apache V2.0.x.

B.2.2.1 Method 1: Building the CSP Module as Shared Object with apxs (APache eXtenSion) Tool

The following command builds and installs the shared library, mod_csp24.so, in the Apache /modules directory using the Apache extension tool, **apxs**. It also adds a directive to load the module to the Apache configuration file /conf/httpd.conf.

```
apxs -I -a –c mod_csp.c
```

This module mod_csp22.so for Apache Version 2.02.x and mod_csp2.so for Apache Version 2.0.x.

Alternatively, mod_csp24.so may be built directly as follows:

```
apxs –c –o mod_csp2.so mod_csp.c
```

Install the shared-object produced binary in the following directory: /opt/cspgateway/bin.

B.2.2.2 Method 2: Building the CSP Module as Shared Object Manually

Perform the following steps to manually build the CSP module as a shared object:

1. Install the module source file mod_csp.c in the following directory: /usr/apache/src/modules/extra
2. Return to the /usr/apache/src directory and edit the Configuration file. Near the end of this file, locate the following line:

   ```
   # AddModule modules/example/mod_example.o
   ```

   After this line, add the following line:

   ```
   ShareModule modules/extra/mod_csp24.so
   ```

3. Configure the build process using the following command:

   ```
   ./Configure
   ```

4. Build the shared object using the following command:

   ```
   make
   ```

5. To produce shared object mod_csp.so in /usr/apache/src/modules/extra

Note: For further information about the `apxs` tool, refer to the Apache documentation at http://httpd.apache.org/docs/2.0/programs/apxs.html.

B.2.2.3 Method 2 Examples

This section documents the compiler and linker commands you can use to build the Apache module on a range of popular UNIX® systems.

- DEC UNIX® 5 (DEC Compiler)

  ```
  cc -c -DOSF1 -std1 -pthread -DIS_64 -ieee_with_inexact \
   –I/usr/apache/include mod_csp.c -o mod_csp.o
  ld -all -shared -expect_unresolved "*" -taso mod_csp.o \
  -o mod_csp24.so
  ```

- FreeBSD (GNU Compiler)

```
cc -c -DFREEBSD –I/usr/apache/include -o mod_csp.o mod_csp.c
ld -G -o mod_csp.so mod_csp24.o
```

- HP-UX (HP Compiler)

```
cc -c -DHPUX11 -DNET_SSL -D_HPUX_SOURCE -Ae +DAportable +z  -DMCC_HTTPD -DSPAPI20
–I/usr/apache/include \
 mod_csp.c -o mod_csp.o
ld -b mod_csp.o -o mod_csp24.so
```

- HP-UX: 64-bit (HP Compiler)

```
cc -c -DHPUX11 -DNET_SSL -D_HPUX_SOURCE -Ae +DAportable +z  -DMCC_HTTPD -DSPAPI20
-I/usr/apache/include \
mod_csp.c -o mod_csp.o
ld -b mod_csp.o -o mod_csp24.so
```

- HP-UX: Itanium (HP Compiler)

```
cc -c -c -DHPUX11 -DNET_SSL -D_HPUX_SOURCE -Ae +z -DMCC_HTTPD  -DSPAPI20 –I/usr/apache/include
mod_csp.c -o mod_csp.o
ld -b mod_csp.o -o mod_csp24.so
```

- Linux: (GNU Compiler)
- Apache v2.0.x

```
cc -c -fpic -DLINUX –I/usr/apache/include -o mod_csp.o mod_csp.c
ld -G -lrt -ldl -o mod_csp.so mod_csp.o
```

- Apache v24.x

```
cc -c -fpic -DLINUX=2 -D_REENTRANT -D_GNU_SOURCE -D_LARGEFILE64_SOURCE –I/usr/apache/include
mod_csp.c -o mod_csp24.o
ld -G mod_csp24.o -lrt -ldl -o mod_csp24.so
```

- Mac OS X: (Mac OS X Compiler)

```
–I/usr/apache/include mod_csp.c -o mod_csp.o
gcc -c -fPIC -fno-common -DMACOSX -DDARWIN gcc -bundle -flat_namespace -undefined suppress mod_csp.o
 -o mod_csp24.so
```

- IBM AIX® (IBM Compiler)

```
xlc_r -c -DAIX -DAIX5 -I/usr/apache/include mod_csp.c -o mod_csp.o
xlc_r -G -H512 -T512 -bM:SRE mod_csp.o -berok -bexpall -bnoentry -lm -lc -o mod_csp24.so
```

- IBM AIX®: 64-bit (IBM Compiler)

```
OBJECT_MODE=64
export OBJECT_MODE
xlc_r -c -DAIX -DAIX5 -I/usr/apache/include mod_csp.c -o mod_csp.o
xlc_r -G -H512 -T512 -bM:SRE mod_csp.o -berok -bexpall -bnoentry -lm -lc -o mod_csp24.so
```

- Sun OS/Solaris: Opteron (Sun Compiler)

```
cc -c -Xa -w -DSOLARIS -KPIC -I/usr/apache/include -o mod_csp.o mod_csp.c
ld -G -o mod_csp.24so mod_csp.o
```

- Sun OS/Solaris: Opteron: 64-bit (Sun Compiler)

```
cc -c -Xa -w -DSOLARIS -KPIC -xarch=amd64 -DBIT64PLAT -I/usr/apache/include -o mod_csp.o mod_csp.c

ld -G -o mod_csp24.so mod_csp.o
```

- Sun OS/Solaris: Intel (Sun Compiler)

```
cc -c -Xa -w -DSOLARIS -KPIC -I/usr/apache/include -o mod_csp.o mod_csp.c
ld -G -o mod_csp24.so mod_csp.o
```

- Sun OS/Solaris: SPARC (Sun Compiler)

```
cc -c -Xa -w -DSOLARIS -I/usr/apache/include
-o mod_csp.o mod_csp.c
old -G -o mod_csp24.so mod_csp.
```

- Sun OS/Solaris: SPAR 64-bit (Sun Compiler)

```
cc -c -Xa -w -DSOLARIS -KPIC -xarch=v9 -DBIT64PLAT -I/usr/apache/include -o mod_csp.o mod_csp.c
ld -G -o mod_csp.24so mod_csp.o
```

Copy mod_csp24.so to: /opt/cspgateway/bin.

B.2.2.4 Runtime Configuration

Edit the Apache configuration file httpd.conf. For the standard Apache distribution, this file is in:

```
/usr/apache/conf
```

For Red Hat Linux, the runtime version of httpd.conf is in:

```
/etc/httpd/conf
```

Assuming that you wish to invoke the CSP engine for requested files that contain a .csp, .cls, or .zen extension, add the following section to the end of httpd.conf. Of course, if you are using Apache Version 2.0.x, specify mod_csp2.so in the configuration block below. If you are using Apache Version 2.2.x, specify mod_csp22.so.

The below configuration block relies on the Regular Expressions (`regex`) processor being available to the Apache environment. Sometimes this is not the case and CSP files are consequently not served (`File not found` errors are returned). To remedy this situation, replace the following lines in the configuration block:

```
<LocationMatch "/*\.([Cc][Ss][Pp]|[Cc][Ll][Ss]|[Zz][En][Nn])$">
                SetHandler csp-handler
</LocationMatch>
```

with:

```
CSPFiletypes csp-handler csp cls

LoadModule csp_module /opt/cspgateway/bin/mod_csp.so
<LocationMatch "/*\.([Cc][Ss][Pp]|[Cc][Ll][Ss]|[Zz][En][Nn])$">
                SetHandler csp-handler
</LocationMatch>
Alias /csp/ /opt/cspgateway/csp/
<Directory "/opt/cspgateway/csp">
        AllowOverride None
        Options MultiViews FollowSymLinks ExecCGI
        Require all granted
        <FilesMatch "\.(log|ini|pid|exe)$">
                Require all denied
        </FilesMatch>
        <Files CSPnsd>
                Require all denied
        </Files>
</Directory>
ScriptAlias /csp-bin/ "/opt/cspgateway/bin/"
ScriptAliasMatch /csp/bin/Systems/Module.cxw "/opt/cspgateway/bin/nph-CSPcgiSys"
ScriptAliasMatch /csp/bin/RunTime/Module.cxw "/opt/cspgateway/bin/nph-CSPcgi"
<Directory "/opt/cspgateway/bin/">
        AllowOverride None
        Options None
        Require all granted
</Directory>
```

Restart Apache after making changes to httpd.conf.

Pooling Connections Between the Apache Module and the NSD

This facility is available only in build 7 (and later) of the Apache module: mod_csp. This is supplied with the Caché v2008.2 distributions.

The enhancement described in this section concerns connection pooling in the Apache module: mod_csp[n].so/dll. As discussed previously, this module is designed to implement the lines of communication between the Web Server environment and the Gateway NSD component (CSPnsd).

Previous versions of the Apache module implemented an HTTP v1.0 style of communication between the Web Server and Gateway. This mode of operation was necessary because, prior to the enhancements made in later versions of the CSP Gateway, CSP responses were generally dispatched to the client without including any form of size notification. Therefore, the "connection close" event was the sole means of terminating a response. However, as with other HTTP v1.0 based applications, the following problems become apparent at times of peak usage:

1. The overhead of creating a new TCP connection to serve each request.
2. Production systems would inevitably experience a build-up of connections spending an unacceptable length of time in the "TIME_WAIT"state at times of peak usage.

The problems listed above can be solved by maintaining a pool of persistent TCP connections between the module and NSD. On the sever side (i.e. NSD), a pool of threads is maintained alongside the persistent connections (connection-oriented thread pooling).

It is anticipated that these enhancements result in a marked improvement in performance for Apache/UNIX® installations. The best improvements are seen with Apache v2 (and later) because this server is a multithreaded/multiprocess hybrid and it is possible to anticipate a single instance of mod_csp being able to maintain a large pool of persistent TCP connections to the server. However, the same optimizations should also improve the performance of the Apache Group's earlier single-threaded/multiprocess servers (v1.3 and earlier).

Controlling Connection Pooling

The size of the connection pool can be controlled by the following Apache configuration parameter (specified in http.conf):

```
CSPMaxPooledNSDConnections <no>
```

In the absence of this parameter, a default value of 32 is used internally – which is effectively:

```
CSPMaxPooledNSDConnections 32
```

To switch-off connection pooling, set this parameter to zero:

```
CSPMaxPooledNSDConnections 0
```

If, for any reason, it becomes necessary to use the legacy (asymmetric) mode of operation (whereby the Gateway notifies the end of response transmission by closing the connection on its side), set this parameter to minus 1:

```
CSPMaxPooledNSDConnections -1
```

Operating and Managing the Gateway with Apache API and NSD

This connectivity option depends on the CSP Gateway's Network Service Daemon (NSD).

1. Start the CSP NSD as described in the section "Operating the Network Service Daemon (NSD)".
2. Restart Apache after making changes to its configuration (httpd.conf).

   The order in which Apache and the NSD are started is unimportant.
3. To access the CSP Web Gateway Management page, enter the following URL in your browser. Although CSP pages are served through the higher-performing module (mod_csp24.so), the CSP Web Gateway Management page is accessed through the CGI module dedicated to this purpose (nph-CSPcgiSys).

   ```
   http://localhost:<port_no>/csp-bin/nph-CSPcgiSys
   ```

If you see an `Unauthorized User` error message, refer to the section on security considerations.

The CSP engine is automatically invoked for requested files that contain a .csp, .cls, or .zen extension. For example:

```
http://localhost:<port_no>/csp/samples/menu.csp
```

B.2.2.5 Registering Additional File Types with CSP

Apache API modules always recognize the following reserved file extensions:

```
.csp .cls .zen .cxw
```

You may have other files that you want to send to CSP for processing. For example, if you need to serve other static files through the CSP Gateway or need to access the Management Portal through this web server, add mappings for file types .jpg, .gif, .png, .css, and .js.

You can configure Apache to recognize what files to pass on to CSP in any of the following ways:

- By `CSP` location directive
- By file extension – `CSPFileTypes` directive
- By MIME type

By CSPlocation directive

Use the CSP directive to request that all files within a certain location be processed by CSP. The following requests that all files and directories under the /csp path be processed by CSP.

```
<Location /csp>
   CSP On
   SetHandler csp-handler-sa
</Location>
```

For example, all the following would be sent to CSP for processing:

```
/csp/
csp/samples/menu.csp
csp/sys/
```

By file extension – CSPFileTypes directive

The `CSPFileTypes` directive works for requests for files that have extensions (such /csp/menu.csp). It does not work for requests for files that do not have file extensions (such as/csp/menu).

This parameter is processed by the Gateway's Apache modules and can be globally defined at the server definition level (in httpd.conf) or restricted within the definition for a location or directory block.

By file type: The following directive requests that files of type xxx and yyy be processed by CSP.

```
CSPFileTypes xxx yyy
```

By location: The following requests that files of type xxx and yyy be processed by CSP but only for locations under /csp (including subdirectories, such as /csp/samples and so on).

```
<Location /csp/>
   CSPFileTypes xxx yyy
</Location>
```

Using the wildcard character, the following requests that all files under path /csp (and /csp/samples and so on) be processed by CSP.

```
<Location /csp/>
   CSPFileTypes *
</Location>
```

By MIME type

In addition to recognizing the file extensions listed above, CSP can also recognize files for the following MIME types:

```
application/x-csp
```

and

```
text/csp
```

For example, to add the file extension xxx to the list of files processed by CSP, use:

```
LoadModule csp_module_sa /opt/cspgateway/csp/bin/CSPa22.so
AddType application/x-csp csp cls zen xxx
```

One of the problems with using MIME types to associate types of file with CSP is that Apache checks to ensure that the path to the resource (that is, the hosting directory) physically exists, and returns a `file not found` error if it does not. It does not, however, check to ensure that the file requested physically exists – which is appropriate for resources served by CSP since they are served by Caché and are virtual as far as the web server is concerned. The "By MIME type" approach is therefore only suitable for cases where the application's path structure can be replicated on the web server.

B.2.3 Alternative Option 2: CGI Modules with NSD (nph-CSPcgi)

The web server should be configured such that it recognizes CSP requests (files of type .csp, .cls, and .zen) and pass them to the CSP gateway for processing.

The web server configuration file (httpd.conf) is found in the following directory:

```
/usr/apache/conf
```

For Red Hat Linux, the runtime version of httpd.conf is found in:

```
/etc/httpd/conf
```

Add the following section to the end of httpd.conf:

```
<LocationMatch "/*\.([Cc][Ss][Pp]|[Cc][Ll][Ss]|[Zz][En][Nn])$">
    AllowOverride None
    Options FollowSymLinks ExecCGI
    Require all granted
</LocationMatch>
ScriptAliasMatch /csp/bin/Systems/Module.cxw "/opt/cspgateway/bin/nph-CSPcgiSys"
ScriptAliasMatch /csp/bin/RunTime/Module.cxw "/opt/cspgateway/bin/nph-CSPcgi"
ScriptAliasMatch /*\.([Cc][Ss][Pp]|[Cc][Ll][Ss])$ "/opt/cspgateway/bin/nph-CSPcgi"
Alias /csp/ instance-installation-directory
<Directory "instance-installation-directory">
    AllowOverride None
    Options MultiViews FollowSymLinks ExecCGI
   Require all granted
    <FilesMatch "\.(log|ini|pid|exe)$">
       Require all denied
    </FilesMatch>
    <Files CSPnsd>
       Require all denied
    </Files>
</Directory>
ScriptAlias /csp-bin/ "/opt/cspgateway/bin/"
<Directory "/opt/cspgateway/bin/">
    AllowOverride None
    Options None
    Require all granted
</Directory>
```

The above configuration block relies on the Regular Expressions (`regex`) processor being available to the Apache environment. Sometimes this is not the case and CSP files are consequently not served (`File not found` errors are returned). To remedy this situation you can associate the (virtual) root location of your CSP applications with the CGI module instead of making the association through the CSP file extensions. For example, your CSP applications are contained in /csp. To associate the CSP CGI module with files under /csp, replace the following configuration block:

```
<LocationMatch "/*\.([Cc][Ss][Pp]|[Cc][Ll][Ss]|[Zz][En][Nn])$">
    AllowOverride None
    Options FollowSymLinks ExecCGI
    Require all granted
</LocationMatch>
ScriptAliasMatch /*\.([Cc][Ss][Pp]|[Cc][Ll][Ss])$ "/opt/cspgateway/bin/nph-CSPcgi"
```

with:

```
<Location "/csp">
    AllowOverride None
    Options FollowSymLinks ExecCGI
   Require all granted
</Location>
ScriptAlias /csp "/opt/cspgateway
/bin/nph-CSPcgi"
```

These directives work for URLs of the form:

```
http://localhost:<port_no>/csp/*.csp
```

Duplicate the configuration block for other root Locations. For example, repeat the process for /myapps for URLs of the form:

```
http://localhost:<port_no>/myapps/*.csp
```

Another approach to avoiding the `regex` issue is to use an `Action` directive in conjunction with a CSP MIME type. However, it should be noted that `Action` is essentially a content filtering technique and, as such, requires that your CSP files are physically present on the web server host even if the Caché server is installed on a separate computer. If you wish to use this approach, first add a new MIME type to the end of the Apache mime.types file and associate it with file types representing CSP content. The mime.types file are found in the same directory as the httpd.conf file.

```
text/csp                csp cls
```

Now, add the `Action` directive to the end of the CGI configuration block in httpd.conf such that it reads:

```
Alias /csp/ /opt/cspgateway/csp/
<Directory "/opt/cspgateway/csp">
    AllowOverride None
    Options MultiViews FollowSymLinks ExecCGI
    Require all granted
<Files CSPnsd>
       Require all denied
    </Files>
<Files CSP.ini>
       Require all denied
    </Files>
<Files CSP.log>
       Require all denied
    </Files>
<Files CSPnsd.ini>
       Require all denied
    </Files>
<Files CSPnsd.pid>
       Require all denied
    </Files>
</Directory>
ScriptAlias /csp-bin/ "/opt/cspgateway/bin/"
<Directory "/opt/cspgateway/bin/">
    AllowOverride None
    Options None
    Require all granted
</Directory>
Action text/csp “/csp-bin/nph-CSPcgi”
```

Restart Apache after making changes to httpd.conf.

Finally, note that because CGI is an open standard, The CSP CGI modules work with any web server.

B.2.3.1 Registering Additional File Types with CSP

Apache API modules always recognize the following reserved file extensions:

```
.csp .cls .zen .cxw
```

You may have other files that you want to send to CSP for processing. For example, if you need to serve other static files through the CSP Gateway or need to access the Management Portal through this web server, add mappings for file types .jpg, .gif, .png, .css, and .js.

You can configure Apache to recognize what files to pass on to CSP in any of the following ways:

- By `CSP` location directive
- By file extension – `CSPFileTypes` directive
- By MIME type

By CSPlocation directive

Use the CSP directive to request that all files within a certain location be processed by CSP. The following requests that all files and directories under the /csp path be processed by CSP.

```
<Location /csp>
   CSP On
   SetHandler csp-handler-sa
</Location>
```

For example, all the following would be sent to CSP for processing:

```
/csp/
csp/samples/menu.csp
csp/sys/
```

By the CSPFileTypes directive

The `CSPFileTypes` directive works for requests for files that have extensions (such /csp/menu.csp). It does not work for requests for files that do not have file extensions (such as/csp/menu).

This parameter is processed by the Gateway's Apache modules and can be globally defined at the server definition level (in httpd.conf) or restricted within the definition for a location or directory block.

By file type: The following directive requests that files of type .xxx and .yyy be processed by CSP.

```
CSPFileTypes xxx yyy
```

By location: The following requests that files of type .xxx and .yyy be processed by CSP but only for locations under /csp (including subdirectories, such as /csp/samples and so on).

```
<Location /csp/>
   CSPFileTypes xxx yyy
</Location>
```

Using the wildcard character, the following requests that all files under path /csp (and /csp/samples and so on) be processed by CSP.

```
<Location /csp/>
   CSPFileTypes *
</Location>
```

By MIME type

In addition to recognizing the file extensions listed above, CSP can also recognize files for the following MIME types:

```
application/x-csp
```

and

```
text/csp
```

For example, to add the file extension xxx to the list of files processed by CSP, use:

```
LoadModule csp_module_sa /opt/cspgateway/csp/bin/CSPa22.so
AddType application/x-csp csp cls zen xxx
```

One of the problems with using MIME types to associate types of file with CSP is that Apache checks to ensure that the path to the resource (that is, the hosting directory) physically exists, and returns a `file not found` error if it does not. It does not, however, check to ensure that the file requested physically exists – which is appropriate for resources served by CSP since they are served by Caché and are virtual as far as the web server is concerned. The "By MIME type" approach is therefore only suitable for cases where the application's path structure can be replicated on the web server.

B.2.3.2 Operating and Managing the Gateway with CGI and NSD

This connectivity option depends on the CSP Gateway's Network Service Daemon (NSD).

1. Start the CSP NSD as described in the section "Operating the Network Service Daemon (NSD)"
2. Restart Apache after making changes to its configuration (httpd.conf).

   The order in which Apache and the NSD are started is unimportant.
3. To access the CSP Web Gateway Management page, enter one of the following URLs in your browser.

   ```
   http://localhost:<port_no>/csp/bin/Systems/Module.cxw
   http://localhost:<port_no>/csp-bin/nph-CSPcgiSys
   ```

If you see an `Unauthorized User` error message, refer to the section on "security considerations".

The CSP engine is automatically invoked for requested files that contain a .csp, .cls, or .zen extension. For example:

```
http://localhost:<port_no>/csp/samples/menu.csp
```

B.2.4 Alternative Option 3: Built-in Apache API Module with NSD (mod_csp.c)

Before embarking on setting up this more complicated option you should bear in mind that, for most modern UNIX® systems, the performance advantage in static linking over linking the module at runtime as a shared object (option 1) is minimal (if anything at all).

Be sure to read these instructions in conjunction with the specific documentation contained within your Apache distribution.

B.2.4.1 Build Apache to Include CSP Module Source Code

Refer to the Apache documentation for this step.

http://httpd.apache.org/

B.2.4.2 Check the Apache Binary Produced

Run the following command to check that the CSP module has been successfully included in the Apache core (this command lists all modules currently built-into Apache):

```
./httpd -l
```

For example:

```
Compiled in modules:
        core.c
        mod_access.c
        mod_auth.c
        mod_include.c
        mod_log_config.c
        mod_env.c
        mod_setenvif.c
        prefork.c
        http_core.c
        mod_mime.c
```

```
        mod_status.c
        mod_autoindex.c
        mod_asis.c
        mod_cgi.c
        mod_negotiation.c
        mod_dir.c
        mod_imap.c
        mod_actions.c
        mod_userdir.c
        mod_alias.c
        mod_csp.c
```

B.2.4.3 Runtime Configuration

Edit the Apache configuration file httpd.conf. For the standard Apache distribution this file is in:

```
/usr/apache/conf
```

For Red Hat Linux, the runtime version of httpd.conf is in:

```
/etc/httpd/conf
```

Assuming that you wish to invoke the CSP engine for requested files that contain a .csp, .cls, or .zen extension, add the following section to the end of httpd.conf:

```
<LocationMatch "/*\.([Cc][Ss][Pp]|[Cc][Ll][Ss]|[Zz][En][Nn])$">
                SetHandler csp-handler
</LocationMatch>
ScriptAliasMatch /csp/bin/Systems/Module.cxw "/opt/cspgateway/bin/nph-CSPcgiSys"
ScriptAliasMatch /csp/bin/RunTime/Module.cxw "/opt/cspgateway/bin/nph-CSPcgi"
Alias /csp/ /opt/cspgateway/csp/
<Directory "/opt/cspgateway/csp">
        AllowOverride None
        Options MultiViews FollowSymLinks ExecCGI
        Require all granted
        <FilesMatch "\.(log|ini|pid|exe)$">
                Require all denied
        </FilesMatch>
        <Files CSPnsd>
                Require all denied
        </Files>
</Directory>
ScriptAlias /csp-bin/ "/opt/cspgateway/bin/"
<Directory "/opt/cspgateway/bin/">
        AllowOverride None
        Options None
        Require all granted
</Directory>
```

The above configuration block relies on the Regular Expressions (`regex`) processor being available to the Apache environment. Sometimes this is not the case and CSP files are consequently not served (`File not found`errors are returned). To remedy this situation, replace the following configuration block:

```
<LocationMatch "/*\.([Cc][Ss][Pp]|[Cc][Ll][Ss]|[Zz][En][Nn])$">
                SetHandler csp-handler
</LocationMatch>
```

with:

```
CSPFiletypes
```

Note that all requests to Apache are serviced by a set of modules invoked in a predefined sequence. The CSP module is one of the first modules invoked, provided its definition was added near the end of the Configuration file as suggested.

Restart Apache after making changes to httpd.conf.

B.2.4.4 Registering Additional File Types with CSP

Apache API modules always recognize the following reserved file extensions:

```
.csp .cls .zen .cxw
```

You may have other files that you want to send to CSP for processing. For example, if you need to serve other static files through the CSP Gateway or need to access the Management Portal through this web server, add mappings for file types .jpg, .gif, .png, .css, and .js.

You can configure Apache to recognize what files to pass on to CSP in any of the following ways:

- By `CSP` location directive
- By file extension
- By `CSPFileTypes` directive
- By MIME type

By CSPlocation directive

Use the CSP directive to request that all files within a certain location be processed by CSP. The following requests that all files and directories under the /csp path be processed by CSP.

```
<Location /csp>
   CSP On
   SetHandler csp-handler-sa
</Location>
```

For example, all the following would be sent to CSP for processing:

```
/csp/
csp/samples/menu.csp
csp/sys/
```

By file extension

Traditionally, Apache has been configured to invoke CSP on the basis of the requested file's extension. For example, the following configuration block (from httpd.conf) tells Apache to allow CSP to process requests for files of type *.csp, *.cls, *.zen, and *.cxw:

```
LoadModule csp_module_sa /opt/cspgateway/csp/bin/CSPa22.so
AddHandler csp-handler-sa csp cls zen cxw
```

This method works well in Apache v1.3.x, but may no longer work in Apache v2.0.x (and later).

By the CSPFileTypes directive

The `CSPFileTypes` directive works for requests for files that have extensions (such /csp/menu.csp). It does not work for requests for files that do not have file extensions (such as/csp/menu).

This parameter is processed by the Gateway's Apache modules and can be globally defined at the server definition level (in httpd.conf) or restricted within the definition for a location or directory block.

By file type: The following directive requests that files of type xxx and yyy be processed by CSP.

```
CSPFileTypes xxx yyy
```

By location: The following requests that files of type xxx and yyy be processed by CSP but only for locations under /csp (including subdirectories, such as /csp/samples and so on).

```
<Location /csp/>
   CSPFileTypes xxx yyy
</Location>
```

Using the wildcard character, the following requests that all files under path /csp (and /csp/samples and so on) be processed by CSP.

```
<Location /csp/>
   CSPFileTypes *
</Location>
```

By MIME type

In addition to recognizing the file extensions listed above, CSP can also recognize files for the following MIME types:

```
application/x-csp
```

and

```
text/csp
```

For example, to add the file extension xxx to the list of files processed by CSP, use:

```
LoadModule csp_module_sa /opt/cspgateway/csp/bin/CSPa22.so
AddType application/x-csp csp cls zen xxx
```

One of the problems with using MIME types to associate types of file with CSP is that Apache checks to ensure that the path to the resource (that is, the hosting directory) physically exists, and returns a `file not found` error if it does not. It does not, however, check to ensure that the file requested physically exists – which is appropriate for resources served by CSP since they are served by Caché and are virtual as far as the web server is concerned. The "By MIME type" approach is therefore only suitable for cases where the application's path structure can be replicated on the web server.

B.2.4.5 Operating and Managing the Gateway with Apache API and NSD

This connectivity option depends on the CSP Gateway's Network Service Daemon (NSD).

1. Start the CSP NSD as described in the section "Operating the Network Service Daemon (NSD)".
2. Restart Apache after making changes to its configuration (httpd.conf).

   The order in which Apache and the NSD are started is unimportant.
3. To access the CSP Web Gateway Management page, point your browser at one of the following locations. Although CSP pages are served through the higher-performing built-in module (mod_csp.c), the CSP Web Gateway Management page is accessed through the CGI module dedicated to this purpose (nph-CSPcgiSys).

   ```
   http://localhost:<port_no>/csp/bin/Systems/Module.cxw
   http://localhost:<port_no>/csp-bin/nph-CSPcgiSys
   ```

If you see an `Unauthorized User` error message, refer to the section on security considerations.

The CSP engine is automatically invoked for requested files that contain a .csp, .cls, or .zen extension. For example:

```
http://localhost:<port_no>/csp/samples/menu.csp
```

B.3 Locked-down Apache for UNIX®, Linux, and Mac OS X

Occasionally Apache is locked-down so that you cannot easily configure the server to access files outside the Apache file system. For example, this is the case for Security Enhanced Linux (SELinux).

For configurations locked down in this way, the CSP Gateway configurations discussed in previous sections results in HTTP 403 Forbidden error codes being returned on attempting to access CSP resources. The simplest way to work with these secure configurations is to copy the file system under:

/opt/cspgateway/csp

to a location under the Apache root:

/usr/apache/csp

Having done this, the appropriate changes to the paths specified in the Apache configuration must be made.

An alternative approach (and the one that should be used if the method suggested above fails) is to configure the CSP Gateway to work within the directories supplied by Apache.

1. Copy CGI modules to: /usr/apache/cgi-bin/

   ```
   cp /opt/cspgateway/bin/*cgi* /usr/apache/cgi-bin/
   ```

2. Copy API modules to: /usr/apache/modules/

   ```
   cp /opt/cspgateway/bin/*.so /usr/apache/modules/
   ```

3. Copy static files (and subdirectories in these directories) to locations under: /usr/apache/htdocs/

   ```
   cp /cache-install-dir/csp/samples/* /usr/apache/htdocs/csp/samples/
   cp /cache-install-dir/csp/broker/* /usr/apache/htdocs/csp/broker
   cp /cache-install-dir/csp/sys/* /usr/apache/htdocs/csp/sys
   ```

4. Install the NSD component (if required) in: /usr/apache/nsd/

Using the directories supplied in Apache simplifies the CSP Gateway configuration in httpd.conf. Modified configuration blocks are shown below.

B.3.1 Recommended Option: Apache API Modules (CSPa24.so)

```
LoadModule cspsys_module_sa /usr/apache/modules/CSPap24.so
CSPSYSModulePath /usr/apache/modules/
<Location "/csp/bin/Systems/">
    SetHandler cspsys-handler-sa
</Location>
<Location "/csp/bin/RunTime/">
    SetHandler csp-handler-sa
</Location>
CSPFiletypes csp cls zen cxw
```

B.3.2 Atypical Option 1: Apache API Module with NSD (mod_csp.so)

```
LoadModule csp_module /usr/apache/modules/mod_csp.so
<LocationMatch "/*\.([Cc][Ss][Pp]|[Cc][Ll][Ss]|[Zz][En][Nn])$">
SetHandler csp-handler
</LocationMatch>
ScriptAliasMatch /csp/bin/Systems/Module.cxw "/usr/apache/cgi-bin/nph-CSPcgiSys"
ScriptAliasMatch /csp/bin/RunTime/Module.cxw "/usr/apache/cgi-bin/nph-CSPcgi"
```

B.3.3 Atypical Option 2: CGI Modules with NSD (nph-CSPcgi)

```
<LocationMatch "/*\.([Cc][Ss][Pp]|[Cc][Ll][Ss]|[Zz][En][Nn])$">
AllowOverride None
Options FollowSymLinks ExecCGI
Require all granted
</LocationMatch>
ScriptAliasMatch /csp/bin/Systems/Module.cxw "/usr/apache/cgi-bin/nph-CSPcgiSys"
ScriptAliasMatch /csp/bin/RunTime/Module.cxw "/usr/apache/cgi-bin/nph-CSPcgi"
ScriptAliasMatch /*\.([Cc][Ss][Pp]|[Cc][Ll][Ss])$ "/usr/apache/cgi-bin/nph-CSPcgi"
```

B.3.4 Atypical Option 3: Built-in Apache API Module with NSD (mod_csp.c)

```
<LocationMatch "/*\.([Cc][Ss][Pp]|[Cc][Ll][Ss]|[Zz][En][Nn])$">
SetHandler csp-handler
</LocationMatch>
ScriptAliasMatch /csp/bin/Systems/Module.cxw "/apache/cgi-bin/nph-CSPcgiSys"
ScriptAliasMatch /csp/bin/RunTime/Module.cxw "/usr/apache/cgi-bin/nph-CSPcgi"
```

B.4 Alternative Option for Sun Web Servers

There is only one atypical option for a Sun web server, the NSAPI module with the NSD.

B.4.1 Install Locations for Sun (Alternative Option)

Install the CSP Gateway components and the CSP static files as follows:

1. The NSD module is:

   ```
   CSPnsd
   ```

   The default location of the NSD module is:

   ```
   /opt/cspgateway/csp
   ```

   The NSD should be run from within its home directory /opt/cspgateway/csp. The configuration file, CSP.INI, and the event log, CSP.LOG, are written in this directory.

   In order to avoid disrupting existing Gateway installations on upgrading Caché, the installation places the following modules in the common location /usr/cspgateway/nsd. This location is not related to a particular Caché instance.

2. NSAPI and CGI Modules

   - CSPcn3.so (NSAPI client to the NSD – if supplied)
   - CSPcgi.exe (Runtime module)
   - nph-CSPcgi.exe (Copy of CSPcgi)
   - CSPcgiSys.exe (Systems-Management module)
   - nph-CSPcgiSys.exe (Copy of CSPcgiSys)

   Note that not all of the modules listed above are required for all connectivity options. Refer to the sections describing each option to see which are actually required.

   The default location for these modules is:

   ```
   /opt/cspgateway/bin
   ```

   The configuration file (CSP.INI) and Event Log (CSP.LOG) are written in this directory for non-NSD based connectivity options.

3. The default location for the HyperEvents components:

   - CSPBroker.js
   - CSPxmlhttp.js

   and miscellaneous static resources (such as image files) are required by the CSP Samples and the Management Portal is:

   cache-install-dir\csp\broker

B.4.2 Alternative Option 1: NSAPI Module with NSD (CSPcn3.so)

In most cases, the all-inclusive NSAPI-based solution (the Recommended Option) is the option of choice, and is the implementation that gives the best performance. The NSAPI/NSD hybrid is useful for cases where it is necessary, for

operational reasons, to manage the Gateway independently of the hosting web server. For example, if multiple instances of the web server are to share the same Gateway installation. In the Recommended Option, each instance of the core web server process binds to its own instance of the Gateway.

This configuration procedure is very similar to that described for the recommended option, "Recommended Option: NSAPI Modules (CSPn3.so)". The essential differences are highlighted below.

Locate the block of core `Init` directives for your web server as described for the Recommended Option. These are found in either magnus.conf or obj.conf. Add the following section before the core `Init` block:

```
Init fn=load-modules shlib=/cache-install-dir/csp/bin/CSPcn3.so funcs=cspc_term,cspc_init,cspc_req
Init fn=cspc_init shlib="/cache-install-dir/csp/bin/CSPcn3.so"
```

B.4.2.1 Directives

Directives for Locating Static Components and the CGI Modules

Find the `default` directives section of obj.conf:

```
<Object name="default">
```

Add the following lines in the `default` section – that is, after the line shown above.

```
NameTrans fn="pfx2dir" from="/csp/samples" dir="/cache-install-dir/csp/samples"
NameTrans fn="pfx2dir" from="/csp/broker" dir="/cache-install-dir/csp/broker"

NameTrans fn="pfx2dir" from="/csp-bin" dir="/cache-install-dir/csp/bin" name="cgi"
```

Directives for Recognizing and Processing CSP Requests

Add the following section to the end of obj.conf:

```
<Object ppath="*/*.csp">
Service method=(GET|HEAD|POST) fn=csp_req
</Object>
<Object ppath="*/*.cls">
Service method=(GET|HEAD|POST) fn=csp_req
</Object>
<Object ppath="*/*.zen">
Service method=(GET|HEAD|POST) fn=csp_req
</Object>
```

B.4.2.2 Operating and Managing the Gateway with NSAPI and NSD

This connectivity option depends on the CSP Gateway's Network Service Daemon (NSD).

1. Start the CSP NSD as described in "Operating the Network Service Daemon (NSD)".
2. Restart the web server after making changes to its configuration files (magnus.conf and obj.conf).

   The order in which the web server and the NSD are started is unimportant.
3. To access the CSP Web Gateway Management page, point your browser at one of the following locations. Although CSP pages are served through the higher-performing module (CSPcn2.so), the CSP Web Gateway Management page is accessed through the CGI module dedicated to this purpose (nph-CSPcgiSys).

   ```
   http://localhost:<port_no>/csp/bin/Systems/Module.cxw
   http://localhost:<port_no>/csp-bin/nph-CSPcgiSys
   ```

If you see an `Unauthorized User` error message, refer to the section on security considerations.

The CSP engine is automatically invoked for requested files that contain a .csp, .cls, or .zen extension. For example:

```
http://localhost:<port_no>/csp/samples/menu.csp
```

B.5 Troubleshooting

If you are getting I/O errors when connecting via HTTPS to an Apache mod_ssl server with Microsoft Internet Explorer, you might try the following. This is an occasional issue caused by Apache introduced in an attempt to work around problems caused by earlier versions of IE. The following lines are in the SSL/TLS configuration for apache:

```
BrowserMatch ".*MSIE.*" nokeepalive ssl-unclean-shutdown downgrade-1.0 force-response-1.0
```

Remove the last \ and the line `downgrade-1.0 force-response-1.0` and restart Apache. This prevents a downgrade of the HTTP headers from HTTP1.1 to HTTP1.0 to get around some issues that earlier versions of IE had with HTTP1.1. This may solve your issue.

C

Apache Considerations UNIX®, Linux, and Mac OS X

This appendix contains information that pertains to the recommended option for UNIX®, Linux, and Mac OS X ("Recommended Option: NSAPI Modules (CSPn3.so)") and atypical option 1 ("Alternative Option 1: Apache API Module with NSD (mod_csp24.so)".

C.1 Apache Process Management and Capacity Planning

Apache provides two process management modules for UNIX-based operating systems. In this architecture, the Gateway modules are directly bound to the Apache *worker processes*. Therefore, the way Apache is configured to manage its process pool has a direct affect on the Gateway.

Apache implements its two process management models as *Multi-Processing Modules* (or MPMs).

Prefork MPM. This is the traditional multi-process (UNIX®) server architecture. It does not use threads and, as a result, there is no requirement that third-party API modules (DSOs) should be thread-safe. Reference: http://httpd.apache.org/docs/2.2/mod/prefork.html

Worker MPM. This is the newer hybrid multithread/multi-process server architecture. It does use threads and all third-party API modules (DSOs) used should be thread-safe. Reference: http://httpd.apache.org/docs/2.2/mod/worker.html.

In order to determine which of the two server models is in use for an existing installation, call the Apache executable directly but qualified as follows:

```
httpd -V
```

The MPM in use will be listed as one of the following:

```
Server MPM:     Prefork
```

Or:

```
Server MPM:     Worker
```

Two further (and related) listings are provided.

`httpd -l` List all modules built-in to the server (including the choice of MPM):

`httpd -L` List all modules and related configuration directives:

The Gateway DSOs are thread-safe and can be deployed in either server model. A useful guide for Apache tuning can be found here:http://httpd.apache.org/docs/2.2/misc/perf-tuning.html

Security. The parent process (of both server architectures) is usually started from an account with superuser privileges assigned (root under UNIX) in order to bind to TCP port 80. The child processes launched by Apache run as a lesser-privileged user. The User and Group directives (in the Apache configuration) are used to set the privileges of the Apache child processes. The child processes must be able to read all the content that they are responsible for serving (and have read/write access to the Gateway's configuration and Event Log files), but, beyond this, should be granted as few privileges as possible.

A brief description of the two Apache MPMs follows. Refer to the Apache documentation (see links above) for further information.

C.1.1 Apache Prefork MPM

The traditional multi-process server architecture.

A single core Apache process (the parent) is responsible for launching child processes. The child processes are responsible for listening on the appropriate TCP port (usually 80), accepting incoming requests from clients, reading the request data and forwarding to whichever module can service the request and generate a response. The latter part of this request processing cycle is in accordance with directives in the Apache configuration file. For example, requests for CSP resources are forwarded to the modules configured for this purpose.

The management of child processes is governed by the following configuration parameters.

- *StartServers* (Defaults to 5): This directive sets the number of child server processes created on startup. As the number of processes is dynamically controlled depending on the load, there is usually little reason to adjust this parameter.

  MinSpareServers (Defaults to 10): The MinSpareServers directive sets the desired minimum number of *idle* child server processes. An idle process is one which is not handling a request. If there are fewer than `MinSpareServers` idle, then the parent process creates new children at a maximum rate of 1 per second.
- *MaxSpareServers* (Defaults to 5): This directive sets the desired maximum number of *idle* child server processes. An idle process is one which is not handling a request. If there are more than `MaxSpareServers` idle, then the parent process kills off the excess processes.
- *MaxClients* (Defaults to 256): This directive sets the limit on the number of simultaneous requests that are served. Any connection attempts over the `MaxClients` limit are normally queued, up to a number based on the `ListenBacklog` directive. Once a child process is freed at the end of a different request, the connection is then serviced. For non-threaded servers, `MaxClients` translates into the maximum number of child processes that are launched to serve requests. To increase this parameter, `ServerLimit` must also be raised.
- *ServerLimit* (Defaults to 5): This directive sets the maximum configured value of `MaxClients` for the lifetime of the Apache instance.
- *MaxRequestsPerChild* (Defaults to 10000): This directive sets the limit on the number of requests that an individual child server process handles. After `MaxRequestsPerChild` requests, the child process dies. If `MaxRequestsPerChild` is 0, then the process never expires.

In general, Apache is self-regulating, so most sites do not need to adjust these directives from their default values. Sites which need to serve more than 256 simultaneous requests may need to increase the value of `MaxClients`, while sites with limited memory may need to decrease `MaxClients` to prevent the server from thrashing (swapping memory to disk and back).

C.1.2 Apache Worker MPM

The hybrid multithread/multi-process server architecture.

A single core Apache process (the parent) is responsible for launching child processes. Each child process creates a fixed number of server threads as specified in the ThreadsPerChild directive, in addition to a listener thread which listens for connections and passes them to a server thread for processing when they arrive.

By using threads to serve requests, the Apache server instance is able to serve a large number of requests with fewer system resources than a process-based server. However, it retains much of the stability of a process-based server by keeping multiple processes available, each with many threads.

Apache always tries to maintain a pool of spare or idle server threads, which stand ready to serve incoming requests. This way, clients do not need to wait for new threads or processes to be created before their requests can be served. The following parameters are provided to control this mechanism.

- *StartServers* (Defaults to 3): This directive sets the number of child server processes created on startup. As the number of processes is dynamically controlled depending on the load, there is usually little reason to adjust this parameter.
- *ThreadsPerChild* (Defaults to 25): This directive sets the number of threads created by each child process.
- *MinSpareThreads* (Defaults to 75): This directive sets the minimum number of idle threads to handle request spikes. If there aren't enough idle threads in the server then child processes are created until the number of idle threads is greater than the number specified here.
- *MaxSpareThreads* (Defaults to 250): This directive sets the maximum number of idle threads. The number of idle threads is considered on a server-wide basis. If there are too many idle threads in the server then child processes are killed until the number of idle threads is less than this number.
- *MaxClients*: This directive restricts the total number of threads that are available to serve clients. The default value is 16 (`ServerLimit`) multiplied by the value of 25 (`ThreadsPerChild`). Therefore, to increase `MaxClients` to a value that requires more than 16 processes, the value of `ServerLimit` must be raised.
- *ServerLimit* (Defaults to 16): Use this directive only if the `MaxClients` and `ThreadsPerChild` settings require more than 16 server processes (default). Do not set the value of this directive any higher than the number of server processes required to honor the requirements specified in the `MaxClients` and `ThreadsPerChild` parameters.
- *ThreadLimit* (Defaults to 64): This directive sets the maximum configured value of `ThreadsPerChild` for the lifetime of the Apache process. Any attempts to change this directive during a restart are ignored, but `ThreadsPerChild` can be modified during a restart up to the value of this directive.
- *MaxRequestsPerChild* (Defaults to 10000): This directive sets the limit on the number of requests that an individual child server process can handle. After `MaxRequestsPerChild` requests, the child process dies and a new process is created (a recycle event). If `MaxRequestsPerChild` is 0, then the process never expires.

The number of child (or worker) processes that are initially launched is set by the `StartServers` directive. Then during operation, Apache assesses the total number of idle threads in all processes, and forks or kills processes to keep this number within the boundaries specified by `MinSpareThreads` and `MaxSpareThreads`. Since this process is self-regulating, it is rarely necessary to modify these directives from their default values. The maximum number of clients that may be served simultaneously (i.e., the maximum total number of threads in all processes) is determined by the `MaxClients` directive. The maximum number of active child processes is determined by the `MaxClients` directive divided by the `ThreadsPerChild` directive.

Two directives set hard limits on the number of active child processes and the number of server threads in a child process, and can only be changed by fully stopping the server and then starting it again. `ServerLimit` is a hard limit on the number of active child processes, and must be greater than or equal to the `MaxClients` directive divided by the `ThreadsPerChild` directive. `ThreadLimit` is a hard limit of the number of server threads, and must be greater than or equal to the `ThreadsPerChild` directive. If non-default values are specified for these directives, they should appear before other worker directives.

In addition to the set of active child processes, there may be additional child processes which are terminating but where at least one server thread is still handling an existing client connection. Up to `MaxClients` terminating processes may be

present, though the actual number can be expected to be much smaller. This behavior can be avoided by disabling the termination of individual child processes as follows:

1. Set the value of `MaxRequestsPerChild` to zero
2. Set the value of `MaxSpareThreads` to the same value as `MaxClients`

C.1.3 Apache MPMs and the Gateway DSOs

The Gateway DSOs are thread-safe and can be deployed in either server model.

For both MPMs the `StartServers` directive specifies the number of child (worker) processes to start. This directive also indicates the number of instances of the CSP Gateway DSOs that can be present – such as one per Apache child process.

Both MPMs involve spreading the load over multiple child (worker) processes and different versions of the Gateway behave differently:

- In Caché version 2010.2 and earlier: Each Gateway instance operates independently. Each instance manages its own running configuration, connection table and form cache. The contents of the Gateway System Status form change with every refresh. This is because the Status displayed is that of the Gateway instance that happens to be returning the result.
- In Caché version 2011.1 and later: Although each Gateway instance is independently loaded by each and every Apache child process, the running configuration, connection table and form cache is held in a shared memory sector. The contents of the Gateway System Status form remain constant with every refresh (apart from changes happening as a result of activity updates, of course). The connection table (and connection numbers) displayed is common to the whole Apache instance and, because of this, an additional column indicating the web server process ID to which each Caché connection is associated is included.

The Gateway load is spread over multiple web servers processes and this has an effect on the following Gateway configuration parameter:

Maximum Server Connections

This parameter allows you to effectively limit the number of connections that the Gateway can make to a particular Caché server. It is a throttle. In setting this parameter bear in mind that the value applies to each web server worker process and not to the web server installation taken as a whole.

For example, if the Maximum Server Connections parameter is set to 10 and the hosting Apache server starts 5 worker processes then the total number of connections that the Gateway can *theoretically* create is 50 (10x5).

This is the simplistic view. However, the effect of the maximum connections throttle is further influenced by the choice of MPM. The Perfork MPM is straightforward: since threads are not used, each Apache worker process (Gateway instance) can only possibly create one connection to Caché. Each worker process can only serve one request at a time. Also bear in mind the effect of the `MaxClients` Apache setting: Apache does not attempt to simultaneously serve more than the number of requests specified here. The Gateway maximum connections throttle, therefore, has no effect when used with the Prefork MPM since it can only have a possible value of one.

The Worker MPM is slightly more complicated since each Apache worker process can start many threads. In theory the total number of connections that can be made to Caché for the whole Gateway installation is the maximum number of server processes (`ServerLimit`) multiplied by the number of threads per process (`ThreadsPerChild`) up to the limits imposed by the `MaxClients` directive. Also bear in mind the ceiling on the number of threads imposed by the `ThreadLimit` setting.

In the Gateway configuration, the limit on the number of connections imposed by the Maximum Server Connections directive applies to each individual Apache worker process. A value higher than the maximum number of threads allowed per process (`ThreadsPerChild`) has little effect since Apache cannot allocate more concurrent work than can be accommodated by the number of threads available in each process. Setting Maximum Server Connections to a value lower

than the number of threads allowed per process (`ThreadsPerChild`) potentially leads to queuing in the Gateway modules since Apache can potentially allocate more work to each worker process than can be handled by the number of allowed connections to Caché. Such a configuration can potentially lead to congestion in the Apache environment so care should therefore be taken.

For installations where most of the Apache workload is made up of CSP requests, it is better to not assign a value to the Gateway's Maximum Server Connections directive and control the amount of concurrent work that can be done (and by implication the number of connections to Caché) with the corresponding Apache configuration parameters.

Setting an independent value for the Gateway's Maximum Server Connections directive would, however, make sense in installations where CSP requests only represent part of the workload for the Apache installation as a whole.

C.2 State-Aware Sessions (Preserve mode 1)

Support for state-aware sessions in a web server that distributes load over multiple worker processes relies on InterProcess Communications (IPC) protocols to manage the routing of requests between individual worker processes. Operating in this web server architecture, the Gateway has no control over which worker process will handle any particular request.

The Gateway uses *UNIX domain sockets* for its IPC protocol and the method for supporting state-aware sessions is described below.

As an example, consider a web server installation that distributes its load over 3 worker processes: P1, P2 and P3. Each worker process can potentially start any number of threads (T1, T2 … Tn) according to the web server MPM and configuration in use.

Suppose an application makes a request to mark its session as state-aware (preserve mode 1) and the Gateway acknowledges this instruction in process P2. The connection and (security context) to the, now private Caché process is hosted by web server worker process P2. All further requests for that user/session must now be processed by worker process P2. However, the Gateway has no control over which worker process the web server will route subsequent requests to so the Gateway must establish an IPC channel between P2 and (potentially) any other worker process in the set.

When the Gateway marks the connection as state-aware in P2 it starts a listening service in a separate, detached, thread. If Gateway Log Level v2 is in use, a message similar to the one shown below is written to the Event Log.

```
IPC Server
Process ID: 28457 Listening on Domain Socket: /tmp/csp28457.str
```

Now, let's say a further request for the same session is processed by worker process P3. The Gateway forwards that request to process P2 via the IPC channel previously established and wait for the response. If Log Level v2 is in use then a message similar to the one shown below is recorded:

```
Route request over IPC to another web server process
PrivateSession=2; pid_self=28456; ipc_to_pid=28457;
```

Of course, if a request for the session happens to be routed by the web server directly to P2 then no further routing is necessary in the Gateway environment since P2 is hosting the session's private connection.

If, for whatever reason, the Gateway is unable to connect and forward a request to a previously created IPC channel, one of the following messages is recorded depending on the context in which the error was raised:

```
IPC CLIENT: Error
Cannot connect
```

Or:

```
IPC CLIENT: Error
Cannot send request
```

The most common reason for problems in this area is if Apache has closed (or recycled) a worker process (in the case of the example: P2). Of course, a process can crash (for example with an access violation/SIGSEGV error) and, in this case, an error message will probably be reported in the Apache error log.

Apache also, by default, periodically recycles worker processes.

If you use state-aware sessions, configure Apache such that it doesn't recycle worker processes by configuring the installation as follows.

- Set the value of `MaxRequestsPerChild` to zero
- Set the value of `MaxSpareThreads` to the same value as MaxClients

If, for whatever reason, it is not possible to prevent Apache periodically recycling processes (perhaps as a result of a mal-functioning module) and state-aware sessions must be used, then an NSD based Gateway configuration can be used. An NSD-based architecture avoids the problems discussed above because it effectively separates the process management of the Gateway from the web server. Options for using the Gateway Network Service Daemon (NSD) are covered in later sections.

D

Building Apache for IBM AIX

Note: It is recommended that you read this whole section through, from beginning to end, before deciding to change the hosting Apache configuration and/or rebuild Apache from source.

This section is relevant to those using the CSP Gateway Dynamic Shared Object (DSO) modules with IBM AIX builds of Apache.

```
CSPa[n][Sys].so
mod_csp[n].so
```

The Apache web server and linked DSOs depend on functionality implemented in two core Apache libraries:

```
libapr
libaprutil
```

These APR libraries are built as part of the main Apache build procedure and contain the functionality responsible for implementing the Apache API (as used by add-on DSO modules such as the CSP Gateway).

On UNIX systems, these libraries are usually built as dynamically linkable shared objects:

```
libapr.so or libapr-1.so
libaprutil.so or libaprutil-1.so
```

The pre-built CSP Gateway DSO modules supplied by InterSystems are built with the expectation that these modules are present as shared objects.

However, under AIX, these libraries are often built as *library archives* and statically linked to the Apache core (`httpd`). This means that they must also be statically linked to add-on DSO modules used by Apache – which is indeed what happens if a DSO is built from source using the Apache Group's `apxs` tool.

The source code to the lightweight module (mod_csp[n].so) intended to be used with the NSD component is supplied both pre-built and as source code. If static linking of the Apache libraries is required then this module can be locally built *in situ*.

The larger stand-alone Gateway modules (CSPa[n][Sys].so) are shipped prebuilt and cannot be used if the hosting Apache server has been statically linked to its core APR libraries. If an attempt is made to use these modules, a fatal error message is reported when Apache is started indicating that functionality provided by the Apache API is missing from the Gateway DSOs. For example:

```
httpd: SyntaSyntax error on line 2 of /usr/local/apache2/conf/trak.conf:
x error on line 411 of /usr/local/apache2/conf/httpd.conf:
Cannot load /opt/cspgateway/bin/CSPapSys.so into server: rtld: 0712-001
Symbol ap_table_set was referenced from module /opt/cspgateway/bin/CSPapSys.so(),
but a runtime definition of the symbol was not found.\nrtld: 0712-001
Symbol ap_table_add was referenced from module /opt/cspgateway/bin/CSPapSys.so(),
but a runtime definition of the symbol was not found.\nrtld: 0712-001
Symbol ap_send_http_header was referenced from module /opt/cspgateway/bin/CSPapSys.so(),
but a runtime definition of the symbol was not found.\nrtld: 0712-001
Symbol ap_table_do was referenced from module /opt/cspgateway/bin/CSPapSys.so(),
but a runtime definition of the symbol was not found.\nrtld: 0712-001
Symbol ap_table_get was referenced from module /opt/cspgateway/bin/CSPapSys.so(),
but a runtime definition of the symbol was not found.\nrtld: 0712-002 fatal error: exiting.
```

The missing symbols are provided by the APR libraries (implemented as shared objects) and these are usually installed in the Apache /lib directory, an example listing of which is shown below:

```
$ ls
apr.exp      libapr-1.so        libaprutil-1.la        libexpat.a
aprutil.exp  libapr-1.so.0      libaprutil-1.so        libexpat.la
libapr-1.a   libapr-1.so.0.4.2  libaprutil-1.so.0      pkgconfig
libapr-1.la  libaprutil-1.a     libaprutil-1.so.0.3.9
```

In the example listing shown above, the APR libraries have been built as shared objects and the Gateway can use them. Check that the path to the shared object's directory is included in the AIX *LIBPATH* environment variable and include it if it is not. For example:

```
export LIBPATH=/usr/local/apache2/lib:$LIBPATH
```

For installations where the APR shared objects are *not* available to the Gateway DSOs, rebuild the Apache core such that it exports (or exposes) the functions (and other symbols) on which the server API depends.

To rebuild Apache proceed as follows:

Unpack the Apache distribution and ensure that the environment is prepared for compilation and linking. For example, to build a 64-bit Apache installation the following environment variables should be set:

```
CFLAGS="-qarch=com -q64"
LDFLAGS="-b64"
OBJECT_MODE=64
export CFLAGS LDFLAGS OBJECT_MODE
```

Apache is usually built using the following three steps, but in this modified procedure there are additional steps to be undertaken between stages (1) and (2).

1. ./configure --enable-so –-prefix=[installation_directory]
2. make
3. make install

Depending on where the final Apache run-time environment is installed, it may be necessary to run the installation phase as SuperUser:

```
sudo make install
```

In order to configure the build process to create an Apache executable exporting all functions/symbols required by non-statically linked third-party DSOs, first run the '*configure'* script as before (step 1 above).

Before running the make command (step 2), find the *Makefile* generated by the *configure* procedure (step 1) and load it into a text editor. This *Makefile* is located in the */server* subdirectory (directly beneath the location in which the build commands are invoked).

Find the lines responsible for creating the exports list (this section is usually found towards the end of the file). For example:

```
# Rule to make exp file for AIX DSOs
httpd.exp: exports.c export_vars.h
    @echo "#! ." > $@
    @echo "* This file was AUTOGENERATED at build time." >> $@
    @echo "* Please do not edit by hand." >> $@
    $(CPP) $(ALL_CPPFLAGS) $(ALL_INCLUDES) exports.c | grep "ap_hack_" | grep -v apr_ | \
           sed -e 's/^.*[)]\(.*\);$$/\1/' >> $@
    $(CPP) $(ALL_CPPFLAGS) $(ALL_INCLUDES) export_vars.h | grep -v apr_ | sed -e 's/^\#[^!]*//' | sed
 -e '/^$$/d' >> $@
```

Now, add the required functions for export directly after the `Please do not edit by hand` line.

Note: Important — each line specifying a function name must begin with a tab character (ASCII 9), not spaces. For convenience, the lines shown below are formatted in this way so you can copy and paste them directly into the target Makefile. Note however, that some lines are too long for this books online production system and are continued with a backslash symbol (\). Remove the backslashes and continue the line in your file.

Example:

```
# Rule to make exp file for AIX DSOs
httpd.exp: exports.c export_vars.h
    @echo "#! ." > $@
    @echo "* This file was AUTOGENERATED at build time." >> $@
    @echo "* Please do not edit by hand." >> $@
    @echo "apr_brigade_cleanup" >> $@
    @echo "apr_brigade_create" >> $@
    @echo "apr_bucket_type_eos" >> $@
    @echo "apr_bucket_type_flush" >> $@
    @echo "apr_create_pool" >> $@
    @echo "apr_global_mutex_unlock" >> $@
    @echo "apr_global_mutex_trylock" >> $@
    @echo "apr_global_mutex_lock" >> $@
    @echo "apr_global_mutex_create" >> $@
    @echo "apr_global_mutex_child_init" >> $@
    @echo "apr_palloc" >> $@
    @echo "apr_pcalloc" >> $@
    @echo "apr_pool_cleanup_null" >> $@
    @echo "apr_pool_cleanup_register" >> $@
    @echo "apr_pool_userdata_get" >> $@
    @echo "apr_pool_userdata_set" >> $@
    @echo "apr_pool_destroy" >> $@
    @echo "apr_pstrcat" >> $@
    @echo "apr_pstrdup" >> $@
    @echo "apr_shm_create" >> $@
    @echo "apr_shm_destroy" >> $@
    @echo "apr_shm_attach" >> $@
    @echo "apr_shm_detach" >> $@
    @echo "apr_shm_baseaddr_get" >> $@
    @echo "apr_shm_size_get" >> $@
    @echo "apr_shm_pool_get" >> $@
    @echo "apr_sleep" >> $@
    @echo "apr_table_add" >> $@
    @echo "apr_table_addn" >> $@
    @echo "apr_table_do" >> $@
    @echo "apr_table_get" >> $@
    @echo "apr_table_make" >> $@
    @echo "apr_table_set" >> $@
    @echo "apr_time_now" >> $@
 $(CPP) $(ALL_CPPFLAGS) $(ALL_INCLUDES) exports.c | grep "ap_hack_" | grep -v apr_ | \
    sed -e 's/^.*[)]\(.*\);$$/\1/' >> $@
 $(CPP) $(ALL_CPPFLAGS) $(ALL_INCLUDES) export_vars.h | grep -v apr_ | \
    sed -e 's/^\#[^!]*//' | sed -e '/^$$/d' >> $@
```

The list shown above includes the bare minimum set of functions/symbols required by the CSP Gateway DSOs. The full list is shown below and can be used if there are plans to use other pre-built third party DSOs requiring functionality over and above that required by the Gateway DSOs.

```
# Rule to make exp file for AIX DSOs
httpd.exp: exports.c export_vars.h
    @echo "#! ." > $@
    @echo "* This file was AUTOGENERATED at build time." >> $@
    @echo "* Please do not edit by hand." >> $@
    @echo "apr_brigade_cleanup" >> $@
    @echo "apr_brigade_create" >> $@
    @echo "apr_bucket_type_eos" >> $@
    @echo "apr_bucket_type_flush" >> $@
    @echo "apr_create_pool" >> $@
```

```
    @echo "apr_global_mutex_unlock" >> $@
    @echo "apr_global_mutex_trylock" >> $@
    @echo "apr_global_mutex_lock" >> $@
    @echo "apr_global_mutex_create" >> $@
    @echo "apr_global_mutex_child_init" >> $@
    @echo "apr_palloc" >> $@
    @echo "apr_pcalloc" >> $@
    @echo "apr_pool_cleanup_null" >> $@
    @echo "apr_pool_cleanup_register" >> $@
    @echo "apr_pool_userdata_get" >> $@
    @echo "apr_pool_userdata_set" >> $@
    @echo "apr_pool_destroy" >> $@
    @echo "apr_pstrcat" >> $@
    @echo "apr_pstrdup" >> $@
    @echo "apr_shm_create" >> $@
    @echo "apr_shm_destroy" >> $@
    @echo "apr_shm_attach" >> $@
    @echo "apr_shm_detach" >> $@
    @echo "apr_shm_baseaddr_get" >> $@
    @echo "apr_shm_size_get" >> $@
    @echo "apr_shm_pool_get" >> $@
    @echo "apr_sleep" >> $@
    @echo "apr_table_add" >> $@
    @echo "apr_table_addn" >> $@
    @echo "apr_table_do" >> $@
    @echo "apr_table_get" >> $@
    @echo "apr_table_make" >> $@
    @echo "apr_table_set" >> $@
    @echo "apr_time_now" >> $@
    $(CPP) $(ALL_CPPFLAGS) $(ALL_INCLUDES) exports.c | grep "ap_hack_" | grep -v apr_ | \
        sed -e 's/^.*[)]\(.*\);$$/\1/' >> $@
    $(CPP) $(ALL_CPPFLAGS) $(ALL_INCLUDES) export_vars.h | grep -v apr_ | \
        sed -e 's/^\#[^!]*//' | sed -e '/^$$/d' >> $@
```

The list shown above includes the bare minimum set of functions/symbols required by the CSP Gateway DSOs. The full list is shown below and can be used if there are plans to use other pre-built third party DSOs requiring functionality over and above that required by the Gateway DSOs.

```
# Rule to make exp file for AIX DSOs
httpd.exp: exports.c export_vars.h
    @echo "#! ." > $@
    @echo "* File was AUTOGENERATED at build time." >> $@
    @echo "* Please do not edit by hand." >> $@
    @echo "apr_allocator_create" >> $@
    @echo "apr_allocator_destroy" >> $@
    @echo "apr_allocator_alloc" >> $@
    @echo "apr_allocator_free" >> $@
    @echo "apr_allocator_owner_set" >> $@
    @echo "apr_allocator_owner_get" >> $@
    @echo "apr_allocator_max_free_set" >> $@
    @echo "apr_allocator_mutex_set" >> $@
    @echo "apr_allocator_mutex_get" >> $@
    @echo "apr_dso_load" >> $@
    @echo "apr_dso_unload" >> $@
    @echo "apr_dso_sym" >> $@
    @echo "apr_dso_error" >> $@
    @echo "apr_env_get" >> $@
    @echo "apr_env_set" >> $@
    @echo "apr_env_delete" >> $@
    @echo "apr_strerror" >> $@
    @echo "apr_stat" >> $@
    @echo "apr_dir_open" >> $@
    @echo "apr_dir_close" >> $@
    @echo "apr_dir_read" >> $@
    @echo "apr_dir_rewind" >> $@
    @echo "apr_filepath_root" >> $@
    @echo "apr_filepath_merge" >> $@
    @echo "apr_filepath_list_split" >> $@
    @echo "apr_filepath_list_merge" >> $@
    @echo "apr_filepath_get" >> $@
    @echo "apr_filepath_set" >> $@
    @echo "apr_filepath_encoding" >> $@
    @echo "apr_file_open" >> $@
    @echo "apr_file_close" >> $@
    @echo "apr_file_remove" >> $@
    @echo "apr_file_rename" >> $@
    @echo "apr_file_copy" >> $@
    @echo "apr_file_append" >> $@
    @echo "apr_file_eof" >> $@
    @echo "apr_file_open_stder" >> $@r
    @echo "apr_file_open_stdout" >> $@
    @echo "apr_file_open_stdin" >> $@
```

```
@echo "apr_file_read" >> $@
@echo "apr_file_write" >> $@
@echo "apr_file_writev" >> $@
@echo "apr_file_read_full" >> $@
@echo "apr_file_write_full" >> $@
@echo "apr_file_putc" >> $@
@echo "apr_file_getc" >> $@
@echo "apr_file_ungetc" >> $@
@echo "apr_file_gets" >> $@
@echo "apr_file_puts" >> $@
@echo "apr_file_flush" >> $@
@echo "apr_file_dup" >> $@
@echo "apr_file_dup2" >> $@
@echo "apr_file_setaside" >> $@
@echo "apr_file_seek" >> $@
@echo "apr_file_pipe_create" >> $@
@echo "apr_file_namedpipe_create" >> $@
@echo "apr_file_pipe_timeout_get" >> $@
@echo "apr_file_pipe_timeout_set" >> $@
@echo "apr_file_lock" >> $@
@echo "apr_file_unlock" >> $@
@echo "apr_file_name_get" >> $@
@echo "apr_file_data_get" >> $@
@echo "apr_file_data_set" >> $@
@echo "apr_file_printf" >> $@
@echo "apr_file_perms_set" >> $@
@echo "apr_file_attrs_set" >> $@
@echo "apr_file_mtime_set" >> $@
@echo "apr_dir_make" >> $@
@echo "apr_dir_make_recursive" >> $@
@echo "apr_dir_remove" >> $@
@echo "apr_file_info_get" >> $@
@echo "apr_file_trunc" >> $@
@echo "apr_file_flags_get" >> $@
@echo "apr_file_pool_get" >> $@
@echo "apr_file_inherit_set" >> $@
@echo "apr_file_inherit_unset" >> $@
@echo "apr_file_mktemp" >> $@
@echo "apr_temp_dir_get" >> $@
@echo "apr_fnmatch" >> $@
@echo "apr_fnmatch_test" >> $@
@echo "apr_initialize" >> $@
@echo "apr_app_initialize" >> $@
@echo "apr_terminate" >> $@
@echo "apr_terminate2" >> $@
@echo "apr_generate_random_bytes" >> $@
@echo "apr_getopt_init" >> $@
@echo "apr_getopt" >> $@
@echo "apr_getopt_long" >> $@
@echo "apr_global_mutex_create" >> $@
@echo "apr_global_mutex_child_init" >> $@
@echo "apr_global_mutex_lock" >> $@
@echo "apr_global_mutex_trylock" >> $@
@echo "apr_global_mutex_unlock" >> $@
@echo "apr_global_mutex_destroy" >> $@
@echo "apr_global_mutex_pool_get" >> $@
@echo "apr_hash_make" >> $@
@echo "apr_hash_copy" >> $@
@echo "apr_hash_set" >> $@
@echo "apr_hash_get" >> $@
@echo "apr_hash_first" >> $@
@echo "apr_hash_next" >> $@
@echo "apr_hash_this" >> $@
@echo "apr_hash_count" >> $@
@echo "apr_hash_overlay" >> $@
@echo "apr_hash_merge" >> $@
@echo "apr_hash_pool_get" >> $@
@echo "apr_filepath_name_get" >> $@
@echo "apr_vformatter" >> $@
@echo "apr_password_get" >> $@
@echo "apr_mmap_create" >> $@
@echo "apr_mmap_dup" >> $@
@echo "apr_mmap_delete" >> $@
@echo "apr_mmap_offset" >> $@
@echo "apr_socket_create" >> $@
@echo "apr_socket_shutdown" >> $@
@echo "apr_socket_close" >> $@
@echo "apr_socket_bind" >> $@
@echo "apr_socket_listen" >> $@
@echo "apr_socket_accept" >> $@
@echo "apr_socket_connect" >> $@
@echo "apr_sockaddr_info_get" >> $@
@echo "apr_getnameinfo" >> $@
@echo "apr_parse_addr_port" >> $@
@echo "apr_gethostname" >> $@
```

```
	@echo "apr_socket_data_get" >> $@
	@echo "apr_socket_data_set" >> $@
	@echo "apr_socket_send" >> $@
	@echo "apr_socket_sendv" >> $@
	@echo "apr_socket_sendto" >> $@
	@echo "apr_socket_recvfrom" >> $@
	@echo "apr_socket_sendfile" >> $@
	@echo "apr_socket_recv" >> $@
	@echo "apr_socket_opt_set" >> $@
	@echo "apr_socket_timeout_set" >> $@
	@echo "apr_socket_opt_get" >> $@
	@echo "apr_socket_timeout_get" >> $@
	@echo "apr_socket_atmark" >> $@
	@echo "apr_socket_addr_get" >> $@
	@echo "apr_sockaddr_ip_get" >> $@
	@echo "apr_sockaddr_equal" >> $@
	@echo "apr_getservbyname" >> $@
	@echo "apr_ipsubnet_create" >> $@
	@echo "apr_ipsubnet_test" >> $@
	@echo "apr_socket_protocol_get" >> $@
	@echo "apr_socket_inherit_set" >> $@
	@echo "apr_socket_inherit_unset" >> $@
	@echo "apr_poll" >> $@
	@echo "apr_pollset_create" >> $@
	@echo "apr_pollset_destroy" >> $@
	@echo "apr_pollset_add" >> $@
	@echo "apr_pollset_remove" >> $@
	@echo "apr_pollset_poll" >> $@
	@echo "apr_pool_initialize" >> $@
	@echo "apr_pool_terminate" >> $@
	@echo "apr_pool_create_ex" >> $@
	@echo "apr_pool_create_ex_debug" >> $@
	@echo "apr_pool_allocator_get" >> $@
	@echo "apr_pool_clear" >> $@
	@echo "apr_pool_clear_debug" >> $@
	@echo "apr_pool_destroy" >> $@
	@echo "apr_pool_destroy_debug" >> $@
	@echo "apr_palloc" >> $@
	@echo "apr_palloc_debug" >> $@
	@echo "apr_pcalloc_debug" >> $@
	@echo "apr_pool_abort_set" >> $@
	@echo "apr_pool_abort_get" >> $@
	@echo "apr_pool_parent_get" >> $@
	@echo "apr_pool_is_ancestor" >> $@
	@echo "apr_pool_tag" >> $@
	@echo "apr_pool_userdata_set" >> $@
	@echo "apr_pool_userdata_setn" >> $@
	@echo "apr_pool_userdata_get" >> $@
	@echo "apr_pool_cleanup_register" >> $@
	@echo "apr_pool_cleanup_kill" >> $@
	@echo "apr_pool_child_cleanup_set" >> $@
	@echo "apr_pool_cleanup_run" >> $@
	@echo "apr_pool_cleanup_null" >> $@
	@echo "apr_pool_cleanup_for_exec" >> $@
	@echo "apr_os_global_mutex_get" >> $@
	@echo "apr_os_file_get" >> $@
	@echo "apr_os_dir_get" >> $@
	@echo "apr_os_sock_get" >> $@
	@echo "apr_os_proc_mutex_get" >> $@
	@echo "apr_os_exp_time_get" >> $@
	@echo "apr_os_imp_time_get" >> $@
	@echo "apr_os_shm_get" >> $@
	@echo "apr_os_thread_get" >> $@
	@echo "apr_os_threadkey_get" >> $@
	@echo "apr_os_thread_put" >> $@
	@echo "apr_os_threadkey_put" >> $@
	@echo "apr_os_thread_current" >> $@
	@echo "apr_os_thread_equal" >> $@
	@echo "apr_os_file_put" >> $@
	@echo "apr_os_pipe_put" >> $@
	@echo "apr_os_pipe_put_ex" >> $@
	@echo "apr_os_dir_put" >> $@
	@echo "apr_os_sock_put" >> $@
	@echo "apr_os_sock_make" >> $@
	@echo "apr_os_proc_mutex_put" >> $@
	@echo "apr_os_imp_time_put" >> $@
	@echo "apr_os_exp_time_put" >> $@
	@echo "apr_os_shm_put" >> $@
	@echo "apr_os_dso_handle_put" >> $@
	@echo "apr_os_dso_handle_get" >> $@
	@echo "apr_os_default_encoding" >> $@
	@echo "apr_os_locale_encoding" >> $@
	@echo "apr_proc_mutex_create" >> $@
	@echo "apr_proc_mutex_child_init" >> $@
	@echo "apr_proc_mutex_lock" >> $@
```

```
@echo "apr_proc_mutex_trylock" >> $@
@echo "apr_proc_mutex_unlock" >> $@
@echo "apr_proc_mutex_destroy" >> $@
@echo "apr_proc_mutex_cleanup" >> $@
@echo "apr_proc_mutex_lockfile" >> $@
@echo "apr_proc_mutex_name" >> $@
@echo "apr_proc_mutex_defname" >> $@
@echo "apr_proc_mutex_pool_get" >> $@
@echo "apr_shm_create" >> $@
@echo "apr_shm_destroy" >> $@
@echo "apr_shm_attach" >> $@
@echo "apr_shm_detach" >> $@
@echo "apr_shm_baseaddr_get" >> $@
@echo "apr_shm_size_get" >> $@
@echo "apr_shm_pool_get" >> $@
@echo "apr_signal" >> $@
@echo "apr_signal_description_get" >> $@
@echo "apr_strnatcmp" >> $@
@echo "apr_strnatcasecmp" >> $@
@echo "apr_pstrdup" >> $@
@echo "apr_pstrmemdup" >> $@
@echo "apr_pstrndup" >> $@
@echo "apr_pmemdup" >> $@
@echo "apr_pstrcat" >> $@
@echo "apr_pstrcatv" >> $@
@echo "apr_pvsprintf" >> $@
@echo "apr_psprintf" >> $@
@echo "apr_cpystrn" >> $@
@echo "apr_collapse_spaces" >> $@
@echo "apr_tokenize_to_argv" >> $@
@echo "apr_strtok" >> $@
@echo "apr_snprintf" >> $@
@echo "apr_vsnprintf" >> $@


@echo "apr_itoa" >> $@
@echo "apr_ltoa" >> $@
@echo "apr_off_t_toa" >> $@
@echo "apr_strtoi64" >> $@
@echo "apr_atoi64" >> $@
@echo "apr_strfsize" >> $@
@echo "apr_table_elts" >> $@
@echo "apr_is_empty_table" >> $@
@echo "apr_is_empty_array" >> $@
@echo "apr_array_make" >> $@
@echo "apr_array_push" >> $@
@echo "apr_array_pop" >> $@
@echo "apr_array_cat" >> $@
@echo "apr_array_copy" >> $@
@echo "apr_array_copy_hdr" >> $@
@echo "apr_array_append" >> $@
@echo "apr_array_pstrcat" >> $@
@echo "apr_table_make" >> $@
@echo "apr_table_copy" >> $@
@echo "apr_table_clear" >> $@
@echo "apr_table_get" >> $@
@echo "apr_table_set" >> $@
@echo "apr_table_setn" >> $@
@echo "apr_table_unset" >> $@
@echo "apr_table_merge" >> $@
@echo "apr_table_mergen" >> $@
@echo "apr_table_add" >> $@
@echo "apr_table_addn" >> $@
@echo "apr_table_overlay" >> $@
@echo "apr_table_do" >> $@
@echo "apr_table_vdo" >> $@
@echo "apr_table_overlap" >> $@
@echo "apr_table_compress" >> $@
@echo "apr_thread_cond_create" >> $@
@echo "apr_thread_cond_wait" >> $@
@echo "apr_thread_cond_timedwait" >> $@
@echo "apr_thread_cond_signal" >> $@
@echo "apr_thread_cond_broadcast" >> $@
@echo "apr_thread_cond_destroy" >> $@
@echo "apr_thread_cond_pool_get" >> $@
@echo "apr_thread_mutex_create" >> $@
@echo "apr_thread_mutex_lock" >> $@
@echo "apr_thread_mutex_trylock" >> $@
@echo "apr_thread_mutex_unlock" >> $@
@echo "apr_thread_mutex_destroy" >> $@
@echo "apr_thread_mutex_pool_get" >> $@
@echo "apr_threadattr_create" >> $@
@echo "apr_threadattr_detach_set" >> $@
@echo "apr_threadattr_detach_get" >> $@
@echo "apr_thread_create" >> $@
```

```
@echo "apr_thread_exit" >> $@
@echo "apr_thread_join" >> $@
@echo "apr_thread_yield" >> $@
@echo "apr_thread_once_init" >> $@
@echo "apr_thread_once" >> $@
@echo "apr_thread_detach" >> $@
@echo "apr_thread_data_get" >> $@
@echo "apr_thread_data_set" >> $@
@echo "apr_threadkey_private_create" >> $@
@echo "apr_threadkey_private_get" >> $@
@echo "apr_threadkey_private_set" >> $@
@echo "apr_threadkey_private_delete" >> $@
@echo "apr_threadkey_data_get" >> $@
@echo "apr_threadkey_data_set" >> $@
@echo "apr_procattr_create" >> $@
@echo "apr_procattr_io_set" >> $@
@echo "apr_procattr_child_in_set" >> $@
@echo "apr_procattr_child_out_set" >> $@
@echo "apr_procattr_child_err_set" >> $@
@echo "apr_procattr_dir_set" >> $@
@echo "apr_procattr_cmdtype_set" >> $@
@echo "apr_procattr_detach_set" >> $@
@echo "apr_procattr_limit_set" >> $@
@echo "apr_procattr_child_errfn_set" >> $@
@echo "apr_procattr_error_check_set" >> $@
@echo "apr_proc_fork" >> $@
@echo "apr_proc_create" >> $@
@echo "apr_proc_wait" >> $@
@echo "apr_proc_wait_all_procs" >> $@
@echo "apr_proc_detach" >> $@
@echo "apr_proc_other_child_register" >> $@
@echo "apr_proc_other_child_unregister" >> $@
@echo "apr_proc_other_child_alert" >> $@
@echo "apr_proc_other_child_refresh" >> $@
@echo "apr_proc_other_child_refresh_all" >> $@
@echo "apr_proc_kill" >> $@
@echo "apr_pool_note_subprocess" >> $@
@echo "apr_setup_signal_thread" >> $@
@echo "apr_signal_thread" >> $@
@echo "apr_thread_pool_get" >> $@
@echo "apr_thread_rwlock_create" >> $@
@echo "apr_thread_rwlock_rdlock" >> $@
@echo "apr_thread_rwlock_tryrdlock" >> $@
@echo "apr_thread_rwlock_wrlock" >> $@
@echo "apr_thread_rwlock_trywrlock" >> $@
@echo "apr_thread_rwlock_unlock" >> $@
@echo "apr_thread_rwlock_destroy" >> $@
@echo "apr_thread_rwlock_pool_get" >> $@
@echo "apr_time_now" >> $@
@echo "apr_time_ansi_put" >> $@
@echo "apr_time_exp_tz" >> $@
@echo "apr_time_exp_gmt" >> $@
@echo "apr_time_exp_lt" >> $@
@echo "apr_time_exp_get" >> $@
@echo "apr_time_exp_gmt_get" >> $@
@echo "apr_sleep" >> $@
@echo "apr_rfc822_date" >> $@
@echo "apr_ctime" >> $@
@echo "apr_strftime" >> $@
@echo "apr_time_clock_hires" >> $@
@echo "apr_uid_current" >> $@
@echo "apr_uid_name_get" >> $@
@echo "apr_uid_get" >> $@
@echo "apr_uid_homepath_get" >> $@
@echo "apr_gid_name_get" >> $@
@echo "apr_gid_get" >> $@
@echo "apr_version" >> $@
@echo "apr_version_string" >> $@
@echo "apr_month_snames" >> $@
@echo "apr_day_snames" >> $@
@echo "apr_base64_encode_len" >> $@
@echo "apr_base64_encode" >> $@
@echo "apr_base64_encode_binary" >> $@
@echo "apr_base64_decode_len" >> $@
@echo "apr_base64_decode" >> $@
@echo "apr_base64_decode_binary" >> $@
@echo "apr_brigade_create" >> $@
@echo "apr_brigade_destroy" >> $@
@echo "apr_brigade_cleanup" >> $@
@echo "apr_brigade_split" >> $@
@echo "apr_brigade_partition" >> $@
@echo "apr_brigade_length" >> $@
@echo "apr_brigade_flatten" >> $@
@echo "apr_brigade_pflatten" >> $@
@echo "apr_brigade_split_line" >> $@
```

```
@echo "apr_brigade_to_iovec" >> $@
@echo "apr_brigade_vputstrs" >> $@
@echo "apr_brigade_write" >> $@
@echo "apr_brigade_writev" >> $@
@echo "apr_brigade_puts" >> $@
@echo "apr_brigade_putc" >> $@
@echo "apr_brigade_putstrs" >> $@
@echo "apr_brigade_printf" >> $@
@echo "apr_brigade_vprintf" >> $@
@echo "apr_bucket_alloc_create" >> $@
@echo "apr_bucket_alloc_create_ex" >> $@
@echo "apr_bucket_alloc_destroy" >> $@
@echo "apr_bucket_alloc" >> $@
@echo "apr_bucket_free" >> $@
@echo "apr_bucket_setaside_noop" >> $@
@echo "apr_bucket_setaside_notimpl" >> $@
@echo "apr_bucket_split_notimpl" >> $@
@echo "apr_bucket_copy_notimpl" >> $@
@echo "apr_bucket_destroy_noop" >> $@
@echo "apr_bucket_simple_split" >> $@
@echo "apr_bucket_simple_copy" >> $@
@echo "apr_bucket_shared_make" >> $@
@echo "apr_bucket_shared_destroy" >> $@
@echo "apr_bucket_shared_split" >> $@
@echo "apr_bucket_shared_copy" >> $@
@echo "apr_bucket_eos_create" >> $@
@echo "apr_bucket_eos_make" >> $@
@echo "apr_bucket_flush_create" >> $@
@echo "apr_bucket_flush_make" >> $@
@echo "apr_bucket_immortal_create" >> $@
@echo "apr_bucket_immortal_make" >> $@
@echo "apr_bucket_transient_create" >> $@
@echo "apr_bucket_transient_make" >> $@
@echo "apr_bucket_heap_create" >> $@
@echo "apr_bucket_heap_make" >> $@
@echo "apr_bucket_pool_create" >> $@
@echo "apr_bucket_pool_make" >> $@
@echo "apr_bucket_mmap_create" >> $@
@echo "apr_bucket_mmap_make" >> $@
@echo "apr_bucket_socket_create" >> $@
@echo "apr_bucket_socket_make" >> $@
@echo "apr_bucket_pipe_create" >> $@
@echo "apr_bucket_pipe_make" >> $@
@echo "apr_bucket_file_create" >> $@
@echo "apr_bucket_file_make" >> $@
@echo "apr_bucket_file_enable_mmap" >> $@
@echo "apr_date_checkmask" >> $@
@echo "apr_date_parse_http" >> $@
@echo "apr_date_parse_rfc" >> $@
@echo "apr_dbm_open_ex" >> $@
@echo "apr_dbm_open" >> $@
@echo "apr_dbm_close" >> $@
@echo "apr_dbm_fetch" >> $@
@echo "apr_dbm_store" >> $@
@echo "apr_dbm_delete" >> $@
@echo "apr_dbm_exists" >> $@
@echo "apr_dbm_firstkey" >> $@
@echo "apr_dbm_nextkey" >> $@
@echo "apr_dbm_freedatum" >> $@
@echo "apr_dbm_geterror" >> $@
@echo "apr_dbm_get_usednames_ex" >> $@
@echo "apr_dbm_get_usednames" >> $@
@echo "apr_hook_sort_register" >> $@
@echo "apr_hook_sort_all" >> $@
@echo "apr_hook_debug_show" >> $@
@echo "apr_hook_deregister_all" >> $@
@echo "apr_md4_init" >> $@
@echo "apr_md4_set_xlate" >> $@
@echo "apr_md4_update" >> $@
@echo "apr_md4_final" >> $@
@echo "apr_md4" >> $@
@echo "apr_md5_init" >> $@
@echo "apr_md5_set_xlate" >> $@
@echo "apr_md5_update" >> $@
@echo "apr_md5_final" >> $@
@echo "apr_md5" >> $@
@echo "apr_md5_encode" >> $@
@echo "apr_password_validate" >> $@
@echo "apr_dynamic_fn_register" >> $@
@echo "apr_dynamic_fn_retrieve" >> $@
@echo "apr_optional_hook_add" >> $@
@echo "apr_optional_hook_get" >> $@
@echo "apr_queue_create" >> $@
@echo "apr_queue_push" >> $@
@echo "apr_queue_pop" >> $@
```

```
    @echo "apr_queue_trypush" >> $@
    @echo "apr_queue_trypop" >> $@
    @echo "apr_queue_size" >> $@
    @echo "apr_queue_interrupt_all" >> $@
    @echo "apr_queue_term" >> $@
    @echo "apr_reslist_create" >> $@
    @echo "apr_reslist_destroy" >> $@
    @echo "apr_reslist_acquire" >> $@
    @echo "apr_reslist_release" >> $@
    @echo "apr_rmm_init" >> $@
    @echo "apr_rmm_destroy" >> $@
    @echo "apr_rmm_attach" >> $@
    @echo "apr_rmm_detach" >> $@
    @echo "apr_rmm_malloc" >> $@
    @echo "apr_rmm_realloc" >> $@
    @echo "apr_rmm_calloc" >> $@
    @echo "apr_rmm_free" >> $@
    @echo "apr_rmm_addr_get" >> $@
    @echo "apr_rmm_offset_get" >> $@
    @echo "apr_rmm_overhead_get" >> $@
    @echo "apr_sdbm_open" >> $@
    @echo "apr_sdbm_close" >> $@
    @echo "apr_sdbm_lock" >> $@
    @echo "apr_sdbm_unlock" >> $@
    @echo "apr_sdbm_fetch" >> $@
    @echo "apr_sdbm_store" >> $@
    @echo "apr_sdbm_delete" >> $@
    @echo "apr_sdbm_firstkey" >> $@
    @echo "apr_sdbm_nextkey" >> $@
    @echo "apr_sdbm_rdonly" >> $@
    @echo "apr_sha1_base64" >> $@
    @echo "apr_sha1_init" >> $@
    @echo "apr_sha1_update" >> $@
    @echo "apr_sha1_update_binary" >> $@
    @echo "apr_sha1_final" >> $@
    @echo "apr_strmatch_precompile" >> $@
    @echo "apr_uri_port_of_scheme" >> $@
    @echo "apr_uri_unparse" >> $@
    @echo "apr_uri_parse" >> $@
    @echo "apr_uri_parse_hostinfo" >> $@
    @echo "apr_uuid_get" >> $@
    @echo "apr_uuid_format" >> $@
    @echo "apr_uuid_parse" >> $@
    @echo "apr_xlate_open" >> $@
    @echo "apr_xlate_sb_get" >> $@
    @echo "apr_xlate_conv_buffer" >> $@
    @echo "apr_xlate_conv_byte" >> $@
    @echo "apr_xlate_close" >> $@
    @echo "apr_text_append" >> $@
    @echo "apr_xml_parser_create" >> $@
    @echo "apr_xml_parse_file" >> $@
    @echo "apr_xml_parser_feed" >> $@
    @echo "apr_xml_parser_done" >> $@
    @echo "apr_xml_parser_geterror" >> $@
    @echo "apr_xml_to_text" >> $@
    @echo "apr_xml_empty_elem" >> $@
    @echo "apr_xml_quote_string" >> $@
    @echo "apr_xml_quote_elem" >> $@
    @echo "apr_xml_insert_uri" >> $@
    @echo "apr_bucket_type_flush" >> $@
    @echo "apr_bucket_type_eos" >> $@
    @echo "apr_bucket_type_file" >> $@
    @echo "apr_bucket_type_heap" >> $@
    @echo "apr_bucket_type_pool" >> $@
    @echo "apr_bucket_type_pipe" >> $@
    @echo "apr_bucket_type_immortal" >> $@
    @echo "apr_bucket_type_transient" >> $@
    @echo "apr_bucket_type_socket" >> $@
    @echo "apr_hook_global_pool" >> $@
    @echo "apr_global_hook_pool" >> $@
    @echo "apr_hook_debug_enabled" >> $@
    @echo "apr_debug_module_hooks" >> $@
    @echo "apr_hook_debug_current" >> $@
    @echo "apr_current_hooking_module" >> $@
    $(CPP) $(ALL_CPPFLAGS) $(ALL_INCLUDES) exports.c | grep "ap_hack_" | grep -v apr_ | \
        sed -e 's/^.*[)]\(.*\);\$$/\1/' >> $@
    $(CPP) $(ALL_CPPFLAGS) $(ALL_INCLUDES) export_vars.h | grep -v apr_ | \
        sed -e 's/^\#[^!]*//' | sed -e '/^$$/d' >> $@
```

Choose whichever list is appropriate for the installation. If the requirements of the CSP Gateway are the only consideration then the first ‘minimal’ list will be sufficient.

Having modified the Makefile, proceed to the make and installation stages (steps 2 and 3). The Apache installation produced works with the pre-built Gateway DSOs supplied by InterSystems.

E

Using Apache DSOs under HP-UX

This section describes how to resolve a common startup problem that can be encountered when using the Gateway Apache DSOs (CSPa[n][Sys].so) under the HP-UX operating system.

The essential symptom is that Apache fails to start and reports an error message to the console indicating a missing symbol, such as `sem_init` (or similar).

Example:

```
apachectl start
Syntax error on line 1137 of /opt/hpws/apache/conf/httpd.conf:
Cannot load /opt/cspgateway/bin/CSPa2.so into server:
Unresolved symbol: sem_init (code) from /opt/cspgateway/bin/CSPa2.so
```

The Semaphore functions are contained in the standard run-time library, librt.so. This is one of a series of libraries that have to be preloaded when thread-safe libraries (such as the Gateway DSOs) are used. This holds true if the hosting executable (Apache `httpd`in this case) doesn't explicitly link to the affected libraries at startup time.

The run-time library is usually found in the /usr/lib/ directory.

Libraries are set for preloading by including them in the *LD_PRELOAD* environment variable.

For example:

```
export LD_PRELOAD=/usr/lib/librt.sl
apachectl start
```

Or:

```
setenv LD_PRELOAD /usr/lib/librt.sl
apachectl start
```

Alternatively, if sudo is used to start Apache:

```
sudo sh -c "export LD_PRELOAD=/usr/lib/librt.sl; apachectl start"
```

After pre-loading the run-time librtary, Apache should start without reporting any further error conditions.

F

IIS Technical Notes

For those interested who use IIS, this appendix describes application pools, web gardens, and bitness.

F.1 IIS Application Pools and Web Gardens

With IIS Version 6, Microsoft further improved the scalability and resilience of the overall web server environment.

IIS Version 6 delivers web hosting services through an adjustable architecture that can be used to manage server resources with improved stability, efficiency, and performance. IIS separates applications into isolated pools of processes and automatically detects memory leaks, defective processes, and overutilized resources. When problems occur, IIS manages them by shutting down and redeploying faulty resources and connecting faulty processes to analytical tools.

IIS Version 6 can run in either of two mutually exclusive modes of operation:

- *Worker process isolation mode*. This is the default mode of IIS 6.0. It isolates key components of the World Wide Web Publishing Service (WWW service) from the effects of errant applications, and protects applications from each other by using the worker process architecture. Microsoft recommends that worker process isolation mode should be used unless there is a specific compatibility issue that makes the use of IIS 5 isolation mode necessary. Web sites that serve static content, simple ASP applications and CSP applications should be able to move to IIS 6.0 running in worker process isolation mode.
- *IIS 5.0 isolation mode*. With this mode, it is possible to run applications that are incompatible with worker process isolation mode because they were developed specifically for earlier versions of IIS. Applications that run correctly on IIS 5.0 should run correctly on IIS 6.0 in IIS 5.0 isolation mode. It is not necessary to use this mode for CSP applications.

Worker process isolation mode provides better default security for running web applications than IIS 5.0 isolation mode. By default, worker processes run with the Network Service identity. The Network Service account has lower access rights than the default account for IIS 5.0 isolation mode. web applications that run in-process in IIS 5.0 application mode run as Local System. The Local System account can read, execute, and change most of the resources on the computer.

F.1.1 Application Pools

An application pool is a configuration that links one or more applications to a set of one or more worker processes. Because applications in an application pool are separated from other applications by worker process boundaries, an application in one application pool is not affected by problems caused by applications running in other application pools.

By creating new application pools and assigning Web sites and applications to them, it is possible to make the server more efficient and reliable. Applications working through pools are always available, even when a worker process serving a different application develops a fault.

Applications are defined by their path in IIS. For example: /csp

F.1.2 Web Gardens

For even greater reliability, it is possible to configure an application pool to be supported by multiple worker processes. An application pool that uses more than one worker processes is called a web garden. The worker processes in a web garden share the requests that arrive for that particular application pool. If a worker process fails, another worker process can continue to process other requests.

It should be noted that web gardens are different from web farms. A web garden is configured on a single server by specifying multiple worker processes for an application pool. web farms use multiple servers for supporting a web site.

Creating a web garden for an application pool can enhance performance in the following situations:

- Robust processing of requests: When a worker process in an application pool is tied up (for example, when a script engine stops responding), other worker processes can accept and process requests for the application pool.
- Reduced contention for resources: When a web garden reaches a steady state, each new TCP/IP connection is assigned, according to a round-robin scheme, to a worker process in the web garden. This helps smooth out workloads and reduce contention for resources that are bound to a worker process.

F.1.3 Application Pools, Web Gardens, and CSP

Application Pool and Web Garden configurations do not affect the operation of NSD-based Gateway configurations because the ISAPI module communicating with the NSD does not pool any persistent information or other resources (such as connections to Caché). All persistent resources are held in the NSD module. The ISAPI module communicating with the NSD is unaffected by changes in the way it is managed by IIS.

The non-NSD based Gateway configuration (CSPms.dll and CSPmsSys.dll) is more sensitive to changes in the way ISAPI extensions are managed in IIS because the pooling of persistent resources (such as connections to Caché) takes place in the extension itself.

Application pools that are configured to use no more than one worker process have no visible impact on the way the Gateway operates within the context of a single web application path (for example, /csp). However, for configurations where multiple worker processes are used (a Web Garden) the workload for the CSP Gateway is evenly distributed amongst all participating worker processes in the pool. Each worker process manages its own instance of the CSP Gateway modules. This process management architecture does not pose a problem with respect to the way the Gateway operates but the following restrictions must be borne in mind:

- IIS must be restarted in order for changes to the Gateway's configuration to take effect. This must be done by completely restarting the World Wide Web Publishing service from the main Windows Services control panel; not through the Internet Services Manager control panel.
- The Gateway's Systems Management form (System Status) cannot be used to accurately monitor the connections used by CSP applications. At any given time the Systems Status reflects the status for the instance of the Gateway that happens to be attached to the current worker process (that is, the worker process that happens to service the Gateway's request).
- Each CSP application (as defined by the web path to the application) maintains its own pool of persistent connections to Caché. Also, each worker process within an application pool maintains its own pool of persistent connections to Caché. This gearing should be remembered when configuring the maximum and minimum number of connections to Caché that the Gateway uses. These settings apply to each and every Gateway instance in the pool.
- State-aware sessions (preserve mode 1) cannot be used with Web Garden configurations because there is no control over the instance of the Gateway which is used to serve any particular request. The net result is that it's not possible to route state-aware requests to their dedicated Caché processes in these configurations.

It is envisaged that some of these restrictions will be completely or partially lifted in future versions of the CSP Gateway. However, it should be remembered that the NSD-based options are not subject to these restrictions because the Gateway is managed independently of IIS.

Finally, the affect of certain worker process configuration parameters on the non-NSD version of the Gateway should be considered. In particular, the effect of the idle timeout and process recycling facility should be borne in mind.

F.1.4 Idle Timeout for Worker Processes

Often it is necessary to conserve system resources by terminating unused worker processes. It is possible to configure a worker process to gracefully close after a specified period of time. This feature can be used to better manage the resources when the processing load is heavy, when identified applications consistently fall into an idle state, or when new processing space is not available.

When a worker process is terminated, the instance of the Gateway that it manages also close and the pool of connections to Caché held by that Gateway instance is terminated. Of course, additional stateless connections can always be replaced in a way that is transparent to users of a CSP application but state-aware sessions (preserve mode 1) terminate when their hosting connection is closed.

F.1.5 Recycling Worker Processes

IIS can be configured to periodically restart worker processes, so that faulty web applications can be recycled. This facility helps to ensure that application pools remain healthy and that any leaked system resources are recovered.

It is possible to configure worker processes to restart based on elapsed time, number of requests served, scheduled times and on the basis of memory usage.

The effect on the CSP Gateway of closing worker processes was discussed in the previous section (Idle Timeout). The same considerations apply here. Because CSP applications can only interact with the CSP Gateway through carefully managed channels, it is recommended that worker processes supporting the CSP applications should not be recycled.

F.2 Bitness — 32-bit Apps on 64-bit Servers for Windows

Note: This section applies to modules that are loaded into the address space of the hosting web serve: ISAPI Extensions and Native Modules (CSPms[Sys].dll and CSPcms.dll). CGI modules are not affected since they run as a detached process with respect to IIS.

With 64 bit Windows 2003 installations, IIS 6.0 could be configured to either run in 32 bit mode or 64 bit mode. The **Enable32bitAppOnWin64** metabase key could be toggled and all worker process would run in the selected bitness mode. This setting applied to the whole IIS installation (that is, it was globally applied to all Application Pools managed by the server).

With IIS 7.0, the **Enable32bitAppOnWin64** setting has been moved down to the Application Pool level. Therefore it is now possible to set the bitness for a particular Application Pool. In other words, it is possible, within a single server installation, to configure one Application Pool to run native 64-bit applications and another to run 32-bit applications.

To access the bitness setting for an Application Pool, enter the IIS control panel:

1. Select **Application Pools** in the left hand panel.
2. Select the appropriate Application Pool.
3. Select **Advanced Settings** in the right hand panel.

4. The **Advanced Settings** dialogue appears. The **Enable 32-Bit Applications** setting is found in the **General** section. This can be set to `True` or `False`.

Incidentally, this configuration setting can be manipulated at the Windows Command line using the *appcmd* command. For example:

```
appcmd set apppool /apppool.name:DefaultAppPool/enable32bitapponwin64:true
```

This sets the Application Pool `DefaultAppPool` to run in 32 bit mode.

It is also possible to list the Application Pools based on bitness using the *appcmd* command. For example, to list all the application pools running in 64 bit mode use the following command:

```
appcmd list apppools /enable32bitapponwin64:false
```

Finally, since application pools can be run in different bitness modes it is necessary to ensure that Native Modules (and ISAPI extensions) that are loaded by the Application Pool are themselves of the correct bitness for the pool. For example, if the hosting application pool is 64-bit then the 64-bit Gateway modules (such as CSPms[Sys].dll) must be used. If the hosting Application Pool is 32-bit then the 32-bit Gateway modules must be used instead.

The bitness check for individual modules is done via a *preCondition* in the module's web.config file. For the CSP Gateway, this file typically looks something like the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
   <system.webServer>
      <handlers>
         <add name="CSPGateway_All" path="*" verb="*" modules="CSPms" resourceType="Unspecified" \
          preCondition="bitness64" />
      </handlers>
      <security>
         <requestFiltering>
            <hiddenSegments>
               <remove segment="bin" />
            </hiddenSegments>
         </requestFiltering>
      </security>
   </system.webServer>
</configuration>
```

Note the bitness setting in the *precondition* clause. In this case bitness is set to `bitness64` which means that IIS checks for 64-bit Gateway modules operating in a 64-bit Application Pool.

If a 32-bit Application Pool is used then the 32-bit Gateway modules must be used and the *preCondition* set to `bitness32`.

If there is an inconsistency between the modules installed, the precondition clause, and/or the expectations of the hosting Application Pool, IIS returns an error condition similar to the one shown below.

```
Error:
The module(s) assigned to this handler mapping has the following preconditions that are not
present in the handler mapping:
bitness64
Runtime errors may occur if a handler mapping does not have a set of preconditions that are
equally as restrictive as, or more restrictive, than the module(s) assigned to the mapping.
Please ensure that this handler mapping has the correct preconditions, and that the
preconditions are not in conflict.
```

G

Using Caché Server Pages with a Remote Web Server

This appendix describes the following configurations using CSP on a remote web server.

- Configuring the Web Server and CSP Gateway
- Accessing CSP on Multiple Caché Servers
- Configuring Apache Virtual Hosts

G.1 Configuring the Web Server and CSP Gateway

This section discusses how to set up a web server and the CSP Gateway to provide access to a CSP application installed on a remote Caché server. The instructions refer to the web server as *Machine W* and to the computer running Caché as *Machine C*. The setup includes the following procedures:

1. Install the CSP Gateway on the Web Server Machine
2. Configure the CSP Gateway
3. If Serving Static Files from the Web Server
4. Configure Web Server Paths

G.1.1 Install the CSP Gateway on the Web Server Machine

Install the CSP Gateway on the web server machine, *Machine W*, where IIS or Apache is running. See the section "Web Server (CSP) Gateway Installation" in the *Caché Installation Guide* if you need more detailed information. During the installation process, follow these instructions:

1. In the **Setup Type** dialog box, select **Web Server** and select **Next**.
2. Review the installation name, type, and destination directory and, if correct, select **Install**.

This creates the CSP directory structure on *Machine W* and creates virtual directory references for the /CSP and /CSP/Bin files.

G.1.2 Configure the CSP Gateway

Next, adjust the CSP Gateway Configuration on *Machine W*. Although the CSP Gateway configuration information is stored in the csp.ini file, always use the Caché Server Pages Web Gateway Management application to update the configuration:

1. Navigate to the CSP **Web Gateway Management** home page by pointing a browser to:

   http://localhost/csp/bin/Systems/Module.cxw

   (Bookmarking this URI is helpful). This link is for your external web server, not the Private Web Server supplied with Caché.

   Note that the link above is correct if you are on the same system that the web server is running on using port 80. If are trying to access the CSP **Web Gateway Management** home page on one system (local system) from another system (remote system), you will be denied access by default. You can access the home page from a remote system in one of two ways: 1) Connect to the home page on the local system and set the `System Manager` field (under `Default Parameters`) to the IP address of the remote system. or 2) Edit the csp.ini file on the local system and add the line: `System_Manager=`*remote-system-ip-address*

2. Select **Server Access** in the left-hand menu. The Caché installation configures a LOCAL server to connect to the Caché instance on the local machine, *Machine W*.

3. Create a new server to represent the Caché instance running on the remote machine, *Machine C*:

   a. Select **Add Server**.

   b. Enter a name for the server (Machine C for example).

   c. Enter the **TCP/IP Address** and **TCP/IP Port** of the remote Caché server on *Machine C*.

   d. Modify the **Connection Security** settings to match the level of authentication expected by *Machine C* for CSP Gateway connections. See the *CSP Gateway and Security* section in this guide for details.

   e. Select **Save Configuration**.

4. Select **Application Access** in the left-hand menu to associate the path to the CSP application on the remote Caché server, *Machine C*, with the server configuration previously created for *Machine C*. The default paths are predefined for / and /csp.

5. Create a new application path to represent the CSP application running on the remote machine, *Machine C*. You can either copy an existing configuration (such as /csp) or select **Add Application** to manually create a new path configuration. The path you create for the application must match that defined for the application in the Caché instance on *Machine C*.

   For example, the default path to the Management Portal is /csp/sys. If you are creating a new application choose your own path name. For example: /myapp or /csp/myapp. Having created the new path, modify the **Default Server** parameter for the path such that it takes the value of the Caché server configuration that you previously set up for *Machine C*.

6. Finally, save the new path configuration.

G.1.3 If Serving Static Files from the Web Server

If you are planning to serve static files from the web server, create directories on *Machine W* to represent your application path. These directories exist solely to hold static content such as image files. You *do not* have to place any CSP files here; they reside on *Machine C*.

Under the directory *install-dir*\CSP on *Machine W*, create \Samples and \User directories. Also create directories to represent other paths which may contain static components referenced in CSP pages. The example in the previous section, requires you to create a directory for \myapp.

G.1.4 Configure Web Server Paths

The application paths in the previous steps correspond to requests for CSP pages in the equivalent locations. For example:

```
http://domain.com/myapp/login.csp
http://domain.com/csp/myapp/login.csp
http://domain.com/csp/sys/login.csp
```

Inheritance is applied in a hierarchical fashion. Consider the following request:

```
http://domain.com/csp/newapp/login.csp
```

The application path configuration for /csp/newapp is used if it exists. If not, the configuration defined for /csp is used instead.

The CSP Gateway installation procedures configure the hosting web server to recognize the /csp virtual path. Typically, these same settings also apply to directories placed under /csp (/csp/myapp, for example).

If you create a new path (such as in the first example, /myapp), you must configure the web server to recognize this new virtual path. These procedures are different depending on the web server you use. Follow the procedures in the section that applies to your web server:

- Add Virtual Directories to IIS
- Add Aliases to Apache Configurations

G.1.4.1 Add Virtual Directories to IIS

The installation procedure for the CSP Gateway configures the virtual directory /csp for CSP applications. If all of your applications are under this virtual directory (for example, /csp/myapp) and you are not using virtual hosts, you do not need to add virtual directories. The instructions in this section apply specifically to the IIS (Internet Information Services) versions 6 and 7 web server.

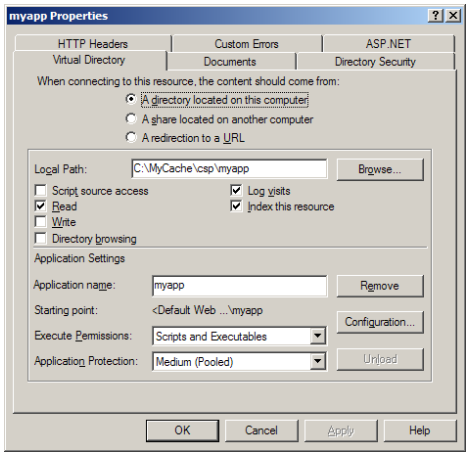
Set up the application path resembling /myapp in the previous examples with properties similar to the /csp virtual directory which is automatically created for you during the CSP Gateway installation.

1. Navigate to the **Internet Information Services** management dialog box, which is likely accessible from the **Administrative Tools** menu of the Windows Control Panel.
2. Expand the folders and right-click **Default Web Site**.
3. Point to **New** and select **Virtual Directory** to create a new directory record with the following values:

   ```
   Alias:                myapp
   Directory:            C:\cache-install-dir\csp\myapp
   Access Permissions:   Select the Execute checkbox (on IIS 6 only)
   ```

4. Either select **Save** and **Apply** all changes, or if you are using the wizard, select **Finish**.

On IIS 6, you can check as follows: Right-click the virtual directory name under **Default Web Site** and select **Properties**. It looks similar to the following figure:



Restart IIS to apply the changes.

G.1.4.2 Add Aliases to Apache Configurations

If you are using an Apache web server to control a remote Caché server and your application path is altered from the /csp default, you must manually add a corresponding alias to the Apache configuration file pointing to the local CSP directory.

For example, to remotely serve the CSP applications on the Caché instance cache-install-dir on *Machine C* from the application path defined on the web server, /myapp/csp, add the following alias line to the httpd.conf file on *Machine W*:

```
Alias /myapp/csp "C:/cache-install-dir/CSP"
```

Restart the Apache web server to apply the changes.

G.2 Accessing CSP on Multiple Caché Servers

Read this section if you need to configure a single web server to access a single CSP application on more than one Caché server. Also read this section if you want to use a single web server to access more than one CSP application on more than one remote Caché server.

This section uses the Management Portal as the example application. Adapt these procedures for your own CSP application.

The Management Portal application is usually called with a URL in this format:

```
http://domain.com/csp/sys/UtilHome.csp
```

Once you have configured your application as described below, then to access this application on different servers, such as servers called cache1 and cache2, you will include the individual Caché servers names as part of the URLs as follows.

```
http://domain.com/cache1/csp/sys/UtilHome.csp
http://domain.com/cache2/csp/sys/UtilHome.csp
```

Changing the Caché Server Name in the URL

You can choose to display the Caché server name in the application path URL or not. If you are content with using the Caché server name in the URL, skip this subsection and proceed to the next subsection called "Configuring the Caché Server for the Application Path".

If you do not want the Caché server name displayed in the CSP application URL, then follow the procedure in this subsection to create a substitute name.

Use the CSPConfigName parameter of the **%System.CSP.SetConfig** method for each of these servers. This example uses linda as the substitute name for server cache1 and perry as the substitute name for a server cache2. Follow this example using your own servers and substitute names.

In a terminal window on the cache1 server, run:

```
d $System.CSP.SetConfig("CSPConfigName","linda")
```

In a terminal window on the cache2 server, run:

```
d $System.CSP.SetConfig("CSPConfigName","perry")
```

1. On cache1, access the CSP Gateway Management page with:

   http://localhost/csp/bin/Systems/Module.cxw

2. Select **Server Access**. Add server configurations for cache1 and cache2. See the section "Configuring Server Access" for details.

3. Select **Application Access**. Create an application path /linda/csp/sys/ with a **Default Server** of cache1. Create an application path /perry/csp/sys/ with a **Default Server** of cache2. See the section "Configuring Application Access" for details.

4. If the web server is IIS then set up virtual directories for /cache1 and /cache2 as described in the *Add Virtual Directories to IIS* section.

   If using an Apache web server see *Add Aliases to Apache Configurations*.

To see other CSP global parameters, enter `%SYS>d $system.CSP.DisplayConfig()`

Configuring the Caché Server for the Application Path

If you are content with using the Caché server name in the URL, follow this procedure. If you do not want the Caché server name displayed in the CSP application URL, then follow the procedure in the preceding subsection "Concealing the Caché Server Name in the URL"

1. On your first server, access the CSP Gateway Management page with:

   http://localhost/csp/bin/Systems/Module.cxw

2. Select **Server Access**. Add server configurations for cache1 and cache2. See the section "Configuring Server Access" for details.

3. Select **Application Access**. Create an application path /cache1/csp/sys/ with a **Default Server** of cache1. Create an application path /cache2/csp/sys/ with a **Default Server** of cache2. See the section "Configuring Application Access" for details.

4. If the web server is IIS then set up virtual directories for /cache1 and /cache2 as described in the *Add Virtual Directories to IIS* section.

   If using an Apache web server see *Add Aliases to Apache Configurations*.

G.3 Configuring Apache Virtual Hosts

An alternative method for accessing an application on multiple servers is to use virtual host arrangements. Virtual hosts are a common feature in Apache web server configurations and are straightforward to set up in this server environment. For example, consider two virtual hosts, each listening on a separate TCP port:

```
http://virtual_host1:81/csp/sys/UtilHome.csp
http://virtual_host2:82/csp/sys/UtilHome.csp
```

Both virtual_host1 and virtual_host2 are served by the same web server and CSP Gateway.

The following shows the Apache configuration (httpd.conf) for this arrangement:

```
<VirtualHost virtual_host1:81>
   ServerName virtual_host1
</VirtualHost>

<VirtualHost virtual_host2:82>
   ServerName virtual_host2
</VirtualHost>
```

Configure the use of these virtual hosts using the CSP Gateway Management application as follows:

1. Navigate to the Caché Server Pages Gateway Management home page by pointing a browser to:

   ```
   http://localhost/csp/bin/Systems/Module.cxw
   ```

2. Select **Server Access** to create a server configuration for cache1 and cache2.
3. Select **Application Access** to create the application paths //virtual_host1/csp/sys/ and //virtual_host2/csp/sys/.

   Note the use of the double forward-slash (//) to introduce the virtual host name.

   Set the **Default Server** for path //virtual_host1/csp/sys/ to be the name of the server configuration set up for cache1 in the previous step.

   Set the **Default Server** for path //virtual_host2/csp/sys/ to be the name of the server configuration set up for cache2 in the previous step.
4. No changes are required in the configuration of the two remote Caché servers. The application path for the portal remains as /csp/sys/ in both cases.

See Virtual Hosts Overview for more information.

G.3.1 Virtual Hosts Overview

Virtual hosts are a means through which you can transparently serve applications on one or more instances through a common web server. Each server installation appears to operate as a separate web server.

The differentiating factor in virtual host setups can be one of the following:

1. Web server IP address — The server hosting the web server is exposed through two IP addresses. For example:

   ```
   123.123.123.1 == www.serverA.com
   123.123.123.2 == www.serverB.com
   ```

2. Web server port — This method is useful for testing different configurations, though it involves including the port number in the request for cases where non-standard TCP ports are used (TCP ports other than 80). For example:

   ```
   Web Server TCP Port 80 == www.serverA.com
   Web Server TCP Port 81 == www.serverB.com
   ```

3. Path — the preferred way of implementing virtual hosts. You register the two names and they translate to a single physical IP address for the web server. For example:

   ```
   www.serverA.com == 123.123.123.1
   www.serverB.com == 123.123.123.1
   ```

Regardless of which way you choose, set up a named slot for each Caché installation in the CSP Gateway configuration (it does not need to be the same as the Caché instance name). The superserver port that the CSP Gateway configuration (for each server) is pointing to is what is important.

For example:

```
www.serverA.com
www.serverB.com
```

Both are served by a single web server installation.

You can implement servers including mixtures of all three. Options 1 and 3 are identical from the browser perspective. You can configure each virtual host to have its own documents root, etc.

To extend the virtual host concept through to CSP, suppose you wish to run the same CSP application through two virtual hosts, but on different Caché instances. For example, one site for testing and another for production.

```
www.serverA.com/csp/login.csp ==> CacheServerA
www.serverB.com/csp/login.csp ==> CacheServerB
```

A CSP application's access to a Caché server is controlled through the CSP Gateway **Application Access** configuration option. Typically, the following two entries are defined:

```
/
/csp
```

The name of the Caché server is associated with these application path definitions:

```
/ (Default Server == CacheServerA)
/csp (Default Server == CacheServerA)
```

The Gateway allows you to extend this configuration to include the name of a virtual host through which you access the application.

```
/ (Default Server == CacheServerA)
/csp (Default Server == CacheServerA)
//www.serverA.com/csp (Default Server == CacheServerA)
//www.serverB.com/csp (Default Server == CacheServerB)
```

You can then configure a separate Caché server for www.serverA.com/csp and www.serverB.com/csp as shown above. Introduce server names by `//`, as shown.

The current rules of inheritance apply. For example, if you request www.serverA.com/xxx/yyy.csp, then the Caché server defined for `/` is ultimately used, unless, you define an ultimate default for serverA as shown below:

```
/ (Default Server == CacheServerL)
/csp (Default Server == CacheServerL)
//www.serverA.com/ (Default Server == CacheServerL)
//www.serverA.com/csp (Default Server == CacheServerA)
//www.serverB.com/csp (Default Server == CacheServerB)
```

Note: The servers specified in the CSP Gateway configuration do not necessarily have to be *virtual*. For example, you can configure a single NSD installation to support several real Apache installations with a different set of Caché servers defined for each one. Further, you can configure each Apache server to support many virtual hosts.

The CSP Gateway identifies the host for the application through the CGI environment variable *SERVER_NAME*.