https://www.iis.net/learn/application-frameworks/install-and-configure-php-applications-on-iis/using-fastcgi-to-host-php-applications-on-iis

Overview

The FastCGI module in IIS enables popular application frameworks that support the FastCGI protocol to be hosted on the IIS Web server in a high performance and reliable way. FastCGI provides a high-performance alternative to the Common Gateway Interface (CGI), which is a standard way of interfacing external applications with Web servers that has been a part of the supported IIS feature set since the first release.

CGI programs are executable files that are launched by the Web server for each request to process the request and generate dynamic responses that are then sent back to the client. Because many of these frameworks do not support multi-threaded execution, CGI enables them to execute reliably on IIS by executing exactly one request per process. Unfortunately, it provides poor performance due to the high cost of starting and shutting down a process for each request.

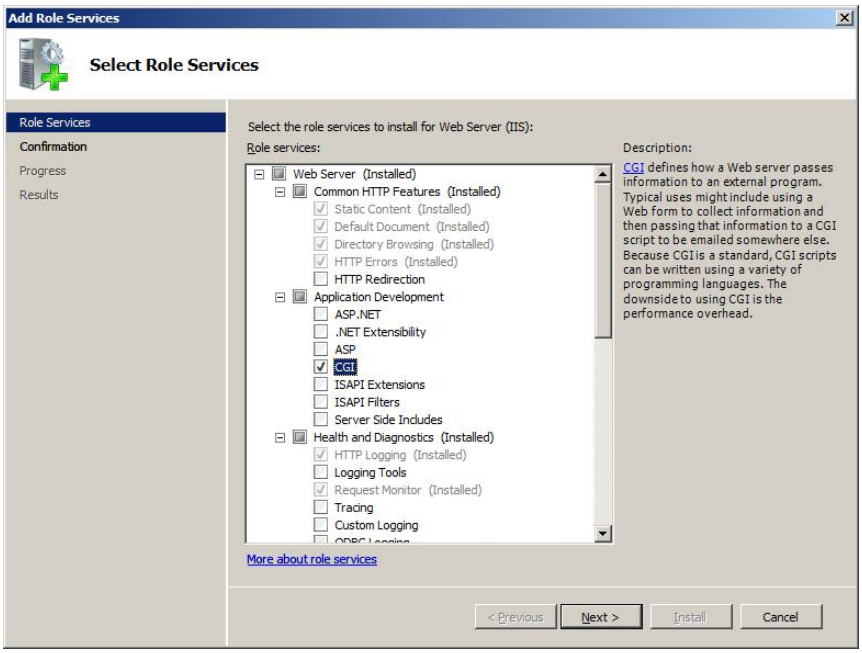
FastCGI addresses the performance issues that are inherent in CGI by providing a mechanism to reuse a single process over and over again for many requests. Additionally, FastCGI maintains compatibility with non-thread-safe libraries by providing a pool of reusable processes and ensuring that each process handles only one request at a time.

reuse a single process over and over again for many requests. Additionally, FastCGI maintains compatibility with non-thread-safe libraries by providing a pool of reusable processes and ensuring that each process handles only one request at a time.

Enable FastCGI Support in IIS

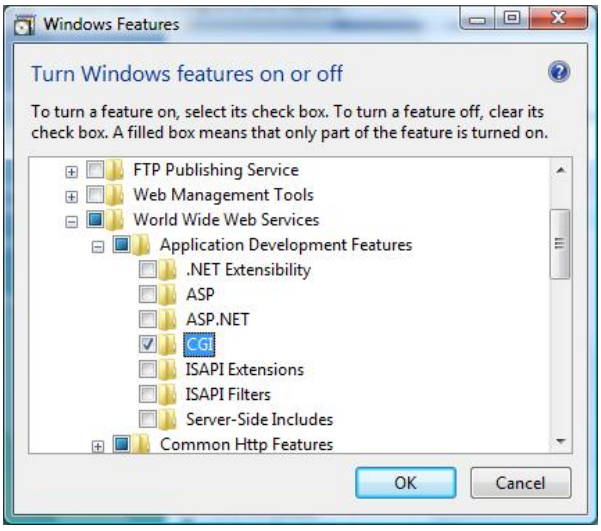
WINDOWS SERVER 2008

Go to **Server Manager** -> **Roles** -> **Add Role Services**. On the **Select Role Services** page, select the **CGI** check box. This enables both the CGI and FastCGI services.



WINDOWS VISTA SP1

Go to **Control Panel** -> **Programs and Features** -> **Turn Windows features on or off**. In the **Windows Features** dialog box, select the **CGI** check box. This enables both the CGI and FastCGI services.



IMPORTANT: INSTALL THE UPDATE FOR THE FASTCGI MODULE

The update for the IIS FastCGI module fixes several known compatibility issues with popular PHP applications. Install the update from one of the following locations:

- Update for Windows Server 2008

applications. Install the update from one of the following locations:

- Update for Windows Server 2008
- Update for Windows Server 2008 x64 Edition
- Update for Windows Server 2008 for Itanium-based Systems
- Update for Windows Vista SP1
- Update for Windows Vista SP1 for x64 based Systems

INSTALL THE ADMINISTRATION PACK FOR IIS

NOTE: This step is optional.

Among other useful features, the Administration Pack for IIS has a convenient user interface for configuring FastCGI settings. The Administration Pack can be installed from the following locations:

- Administration Pack for IIS 7 and Above - x86
- Administration Pack for IIS 7 and Above - x64

Install and Configure PHP

It is recommended that you use a non-thread safe build of PHP with IIS FastCGI. A non-thread safe build of PHP provides significant performance gains over the standard build by not doing any thread-safety checks, which are not necessary, since FastCGI ensures a single threaded execution environment.

To install PHP:

1. Download the latest non-thread safe zip package with binaries of PHP: http://www.php.net/downloads.php.
2. Unpack the files to the directory of your choice (e.g. C:\PHP). Rename the php.ini-recommended file to php.ini.
3. Open the php.ini file. Uncomment and modify the settings as follows:
    - Set **fastcgi.impersonate = 1**. FastCGI under IIS supports the ability to impersonate security tokens of the calling client. This allows IIS to define the security context that the request runs under.
    - Set **cgi.fix_pathinfo=1**. cgi.fix_pathinfo provides *real* PATH_INFO/PATH_TRANSLATED support for CGI. Previously, PHP behavior was to set PATH_TRANSLATED to SCRIPT_FILENAME, and to not define PATH_INFO. For more information about PATH_INFO, see the cgi specifications. Setting this value to 1 will cause PHP CGI to fix its paths to conform to the specifications.
    - Set **cgi.force_redirect** = 0.
    - Set **open_basedir** to point to the folder or network path where the content of the Web site(s) is located.
    - Set **extension_dir** to point to the location where the PHP extensions are located. Typically, for PHP 5.2.X the value would be set as **extension_dir = "./ext"**
    - Enable the required PHP extension by un-commenting the corresponding lines, for example:

      extension=php_mssql.dll
      extension=php_mysql.dll
4. Open a command prompt, and run the following command to verify that PHP installed successfully:

```
C:\PHP>php -info
```

```
C:\PHP>php -info
```

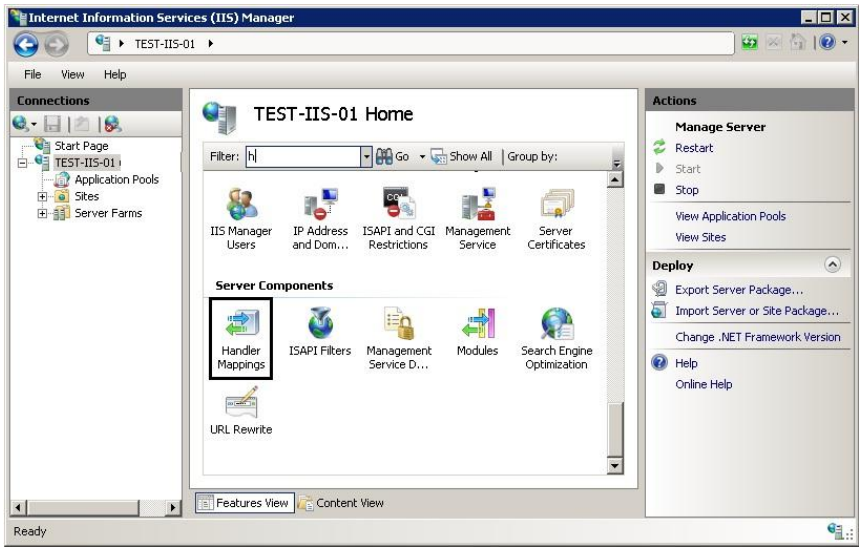
If PHP installed correctly and all its dependencies are available on the machine, this command will output the current PHP configuration information.

Configure IIS to Handle PHP Requests

For IIS to host PHP applications, you must add a handler mapping that tells IIS to pass all PHP-specific requests to the PHP application framework by using the FastCGI protocol.

CONFIGURE IIS TO HANDLE PHP REQUESTS BY USING IIS MANAGER

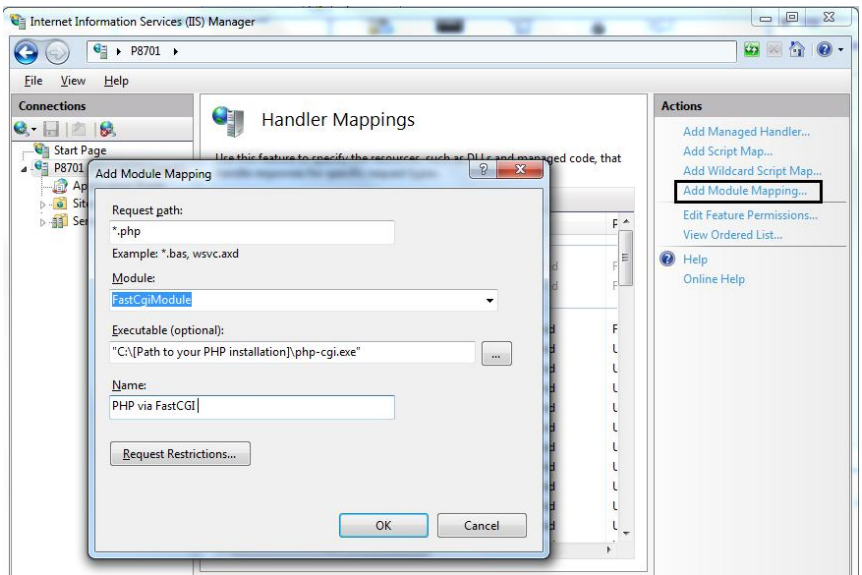
1. Open IIS Manager. At the server level, double-click **Handler Mappings**.

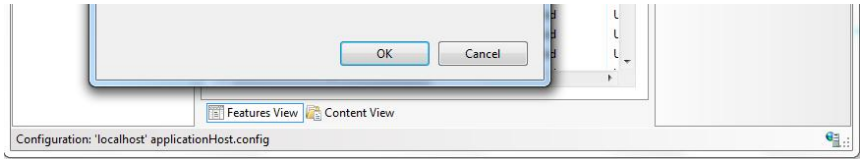


2. In the **Actions** pane, click **Add Module Mapping...**. In the **Add Module Mapping** dialog box, specify the configuration settings as follows:

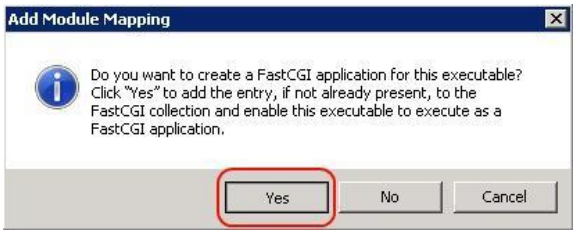
Request path: ***.php**
Module: **FastCgiModule**
Executable: **"C:\[Path to your PHP installation]\php-cgi.exe"**
Name: **PHP via FastCGI**

3. Click **OK**.





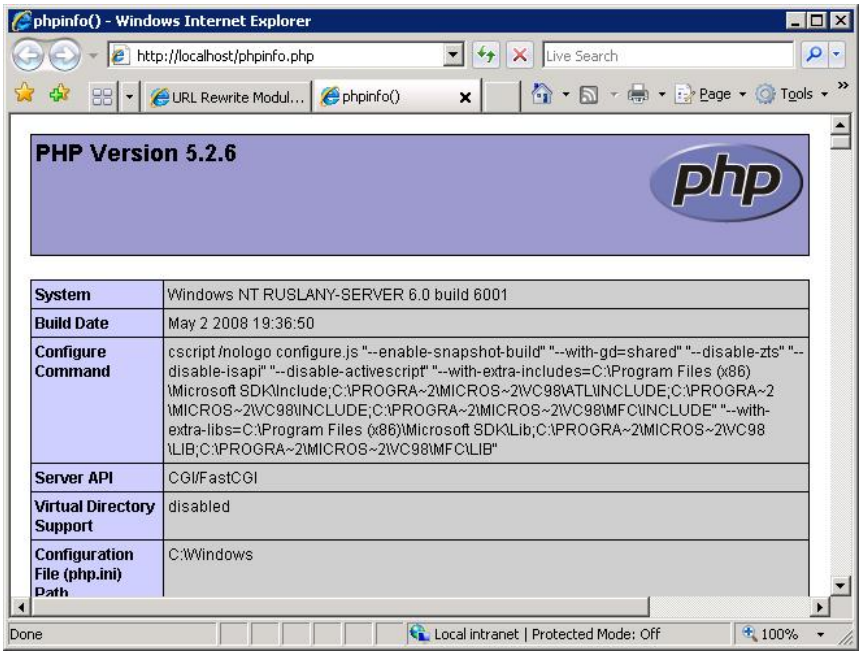
4. In the **Add Module Mapping** confirmation dialog box that asks if you want to create a FastCGI application for this executable, click **Yes**.



5. Test that the handler mapping works correctly by creating a phpinfo.php file in the C:\inetpub \wwwroot folder that contains the following code:

```
<?php phpinfo(); ?>
```

6. Open a browser and navigate to http://localhost/phpinfo.php. If everything was setup correctly, you will see the standard PHP information page.



NOTE: If you do not see **FastCgiModule** in the **Modules:** list, the module is either not registered or not enabled. To check if the FastCGI module is registered, open the IIS configuration file that is located at %windir%\windows\system32\config\applicationHost.config and check that the following line is present in the <globalModules> section:

```
<add name="FastCgiModule" image="%windir%\System32\inetsrv\iisfcgi.dll" />
```

In the same file, also check that the FastCGI module is added to the <modules> section:

```
<add name="FastCgiModule" />
```

CONFIGURE IIS TO HANDLE PHP REQUESTS BY USING THE COMMAND LINE

Alternatively, you can complete the steps above by using the command line tool A**ppCmd**.

1. Create the FastCGI application process pool by running the following command:

1. Create the FastCGI application process pool by running the following command:

```
C:\>%windir%\system32\inetsrv\appcmd set config /section:system.webServer/fas
```

2. Create the handler mapping by running the following command:

```
C:\>%windir%\system32\inetsrv\appcmd set config /section:system.webServer/han
```

Note: If you are using PHP version 4.X, you can use php.exe instead of php-cgi.exe.

Best Practices for Configuring FastCGI and PHP

This download contains a summary presentation on Best Practices for hosting PHP in a shared hosting environment.

SECURITY ISOLATION FOR PHP WEB SITES

The recommendation for isolating PHP Web sites in a shared hosting environment is consistent with all general security isolation recommendations for IIS. In particular, it is recommended to:

- Use one application pool per Web site
- Use a dedicated user account as an identity for the application pool
- Configure an anonymous user identity to use the application pool identity
- Ensure that FastCGI impersonation is enabled in the php.ini file (fastcgi.impersonate=1)

For more details about security isolation in a shared hosting environment, see Ensure Security Isolation for Web Sites.

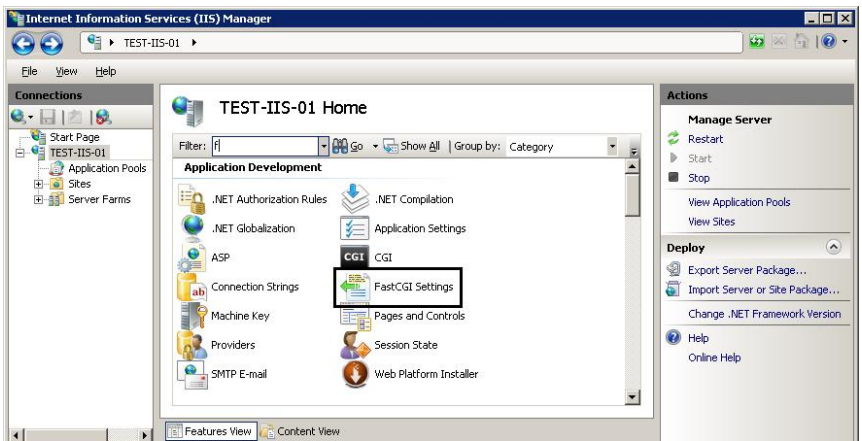
PHP PROCESS RECYCLING BEHAVIOR

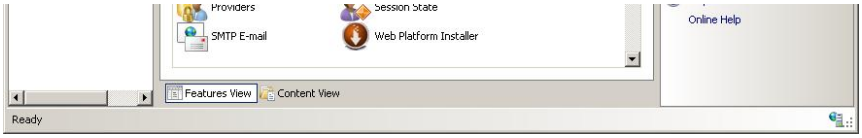
Ensure that FastCGI always recycles the php-cgi.exe processes before the native PHP recycling kicks in. The FastCGI process recycling behavior is controlled by the configuration property **instanceMaxRequests**. This property specifies how many requests the FastCGI process will process before recycling. PHP also has a similar process recycling functionality that is controlled by the environment variable **PHP_FCGI_MAX_REQUESTS**. By setting **instanceMaxRequests** to be less than or equal to **PHP_FCGI_MAX_REQUESTS**, you can ensure that the native PHP process recycling logic will never kick in.

The FastCGI settings can be configured either by using IIS Manager or by using the command line tool A**ppCmd**.

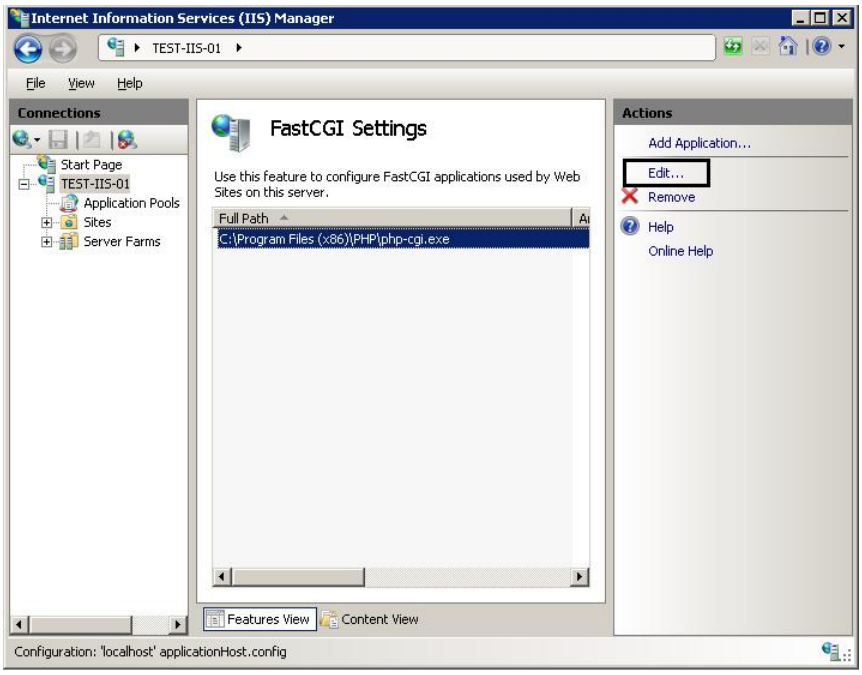
Configure FastCGI recycling settings by using IIS Manager

1. Ensure that the Administration Pack for IIS is installed on your server. Open IIS Manager. On the server level, double-click **FastCGI Settings**.

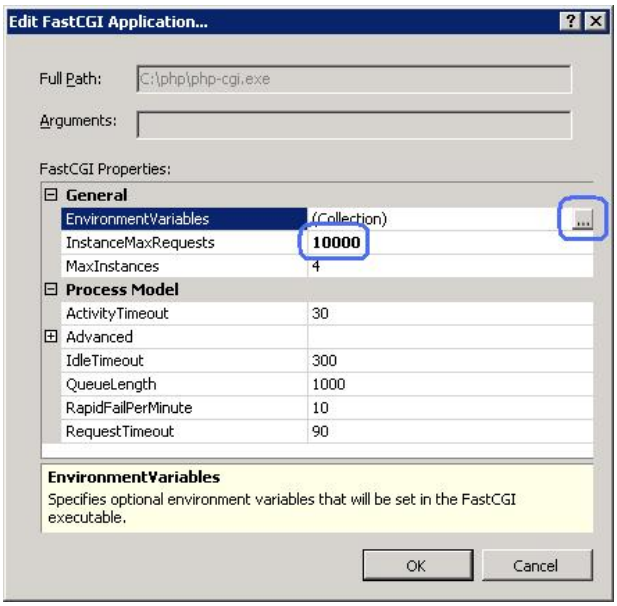




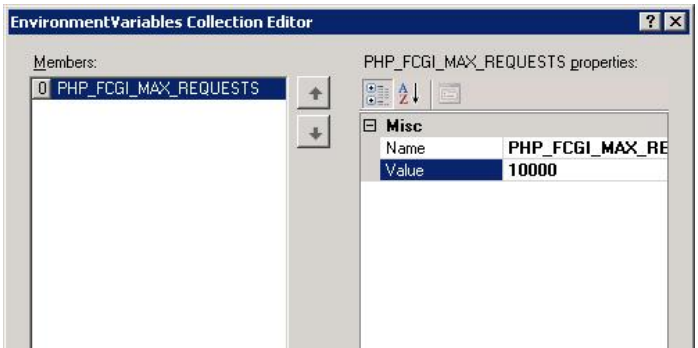
2. Select the FastCGI application that you want to configure. In the **Actions** pane, click **Edit...**.

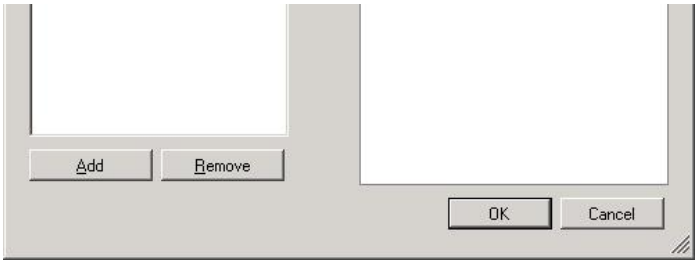


3. In the **Edit FastCGI Application** dialog box, set the **InstanceMaxRequests** to **10000**. Next to the **EnvironmentVariables** setting, click the Browse (**...**) button.



4. In the **EnvironmentVariables Collection Editor** dialog box, add the **PHP_FCGI_MAX_REQUESTS** environment variable and set its value to **10000**.





Note: If you do not configure these settings, the following default settings will be used: **instanceMaxRequests** = 200, **PHP_FCGI_MAX_REQUESTS** = 500 (on most PHP builds).

Configure FastCGI recycling settings by using the command line

Configure the recycling behavior of FastCGI and PHP by using **AppCmd** by running the following commands:

```
C:\>%windir%\system32\inetsrv\appcmd set config -section:system.webServer/fas

C:\>%windir%\system32\inetsrv\appcmd.exe set config -section:system.webServer
```

PHP VERSIONING

Many PHP applications rely on functions or features that are available only in certain versions of PHP. If these types of applications are to be hosted on the same server, different PHP versions must be enabled and running side-by-side. The IIS FastCGI handler fully supports running multiple versions of PHP on the same Web server.

For example, assume that on your Web server you plan to support PHP 4.4.8, PHP 5.2.1, and PHP 5.2.5 non-thread safe. To enable that configuration, you must place corresponding PHP binaries in separate folders on the file system (e.g. C:\php448\, C:\php521\ and C:\php525nts) and then create FastCGI application process pools for each version:

```
C:\>%windir%\system32\inetsrv\appcmd set config /section:system.webServer/fas

C:\>%windir%\system32\inetsrv\appcmd set config /section:system.webServer/fas

C:\>%windir%\system32\inetsrv\appcmd set config /section:system.webServer/fas
```

If you have three Web sites (site1, site2, site3) and each site must use a different PHP version, you can now define handler mappings on each of those sites to reference a corresponding FastCGI application process pool.

Note: Each FastCGI process pool is uniquely identified by a combination of fullPath and arguments properties.

```
C:\>%windir%\system32\inetsrv\appcmd set config site1 -section:system.webServ

C:\>%windir%\system32\inetsrv\appcmd set config site2 -section:system.webServ

C:\>%windir%\system32\inetsrv\appcmd set config site3 -section:system.webServ
```

PHP SECURITY RECOMMENDATIONS

The following settings can be used to tighten the security of a PHP installation. To make the recommended changes, locate and open the php.ini file and edit the configuration settings as described below:

Setting	Description
allow_url_fopen=Off allow_url_include=Off	Disable remote URLs for file handling functions, which may cause code injection vulnerabilities.
register_globals=Off	Disable register_globals.

allow_url_include=Off	code injection vulnerabilities.
register_globals=Off	Disable register_globals.
open_basedir="c:\inetpub\"	Restrict where PHP processes can read and write on a file system.
safe_mode=Off safe_mode_gid=Off	Disable safe mode.
max_execution_time=30 max_input_time=60	Limit script execution time.
memory_limit=16M upload_max_filesize=2M post_max_size=8M max_input_nesting_levels=64	Limit memory usage and file sizes.
display_errors=Off log_errors=On error_log="C:\path\of\your\choice"	Configure error messages and logging.
fastcgi.logging=0	The IIS FastCGI module will fail the request when PHP sends any data on stderr by using the FastCGI protocol. Disable FastCGI logging to prevent PHP from sending error information over stderr and generating 500 response codes for the client.
expose_php=Off	Hide the presence of PHP.

Enabling per-site PHP configuration

This section describes the recommended way of enabling per-site PHP configuration. This recommendation was discovered and validated by Radney Jasmin with hosting provider GoDaddy.com who now offers PHP hosting on Windows Server 2008 by using FastCGI.

PER-SITE PHP PROCESS POOLS

When each Web site has its own application pool, which is a recommended practice on IIS, it is possible to associate a dedicated FastCGI process pool with each Web site. A FastCGI process pool is uniquely identified by the combination of **fullPath** and **arguments** attributes. If you need to create several FastCGI process pools for the same process executable, such as php-cgi.exe, you can use the **arguments** attribute to distinguish the process pool definitions. With php-cgi.exe processes, you can also use the command line switch "-d" to define an INI entry for a PHP process. You can use this switch to set a PHP setting that makes the arguments string unique.

For example, if there are two Web sites "website1" and "website2" that must have their own set of PHP settings, the FastCGI process pools can be defined as follows:

```
<fastCgi>
    <application fullPath="C:\PHP\php-cgi.exe" arguments="-d open_basedir=C:\Webs
    <application fullPath="C:\PHP\php-cgi.exe" arguments="-d open_basedir=C:\Webs
</fastCgi>
```

In this example the PHP setting **open_basedir** is used to distinguish between the process pool definitions. The setting also enforces that the PHP executable for each process pool can perform file operations only within the root folder of the corresponding Web site.

Then website1 can have the PHP handler mapping as follows:

```
<system.webServer>
    <handlers accessPolicy="Read, Script">
        <add name="PHP via FastCGI" path="*.php" verb="*" modules="FastCgiMod
    </handlers>
</system.webServer>
```

and website2 can have the PHP handler mapping as follows:

```
</system.webServer>
```

and website2 can have the PHP handler mapping as follows:

```
<system.webServer>
    <handlers accessPolicy="Read, Script">
        <add name="PHP via FastCGI" path="*.php" verb="*" modules="FastCgiMod
    </handlers>
</system.webServer>
```

SPECIFYING PHP.INI LOCATION

When the PHP process starts, it determines the location of the configuration php.ini file by using various settings. The PHP documentation provides a detailed description of the PHP startup process. One of the places where the PHP process searches for the php.ini location is the PHPRC environment variable. If the PHP process finds a php.ini file in the path that is specified in this environment variable, it will use it; otherwise, the PHP process will revert to using the default location of the php.ini file. This environment variable can be used to allow hosting customers to use their own versions of php.ini files.

For example if there are two Web sites "website1" and "website2" that are located at the following file paths: C:\WebSites\website1 and C:\WebSites\website2, you can configure the php-cgi.exe process pools in the <fastCgi> section of the applicationHost.config file as follows:

```
<fastCgi>
    <application fullPath="C:\PHP\php-cgi.exe" arguments="-d open_basedir=C:\
        <environmentVariables>
            <environmentVariable name="PHPRC" value="C:\WebSites\website1" />
        </environmentVariables>
    </application>
    <application fullPath="C:\PHP\php-cgi.exe" arguments="-d open_basedir=C:\
        <environmentVariables>
            <environmentVariable name="PHPRC" value="C:\WebSites\website2" />
        </environmentVariables>
    </application>
</fastCgi>
```

This way website1 can have its own version of the php.ini file that is located in the C:\WebSites \website1, while website2 can have its own version of the php.ini file that is located in C:\WebSites \website2. This configuration also ensures that if a php.ini file cannot be found in the location that is specified by the PHPRC environment variable, then PHP will use the default php.ini file that is located in the same folder where the php-cgi.exe is located.