# File Change Notifications (FCN) in ASP.NET Site Performance

IIS runs any ASP.NET web site from a specific location that can be stored locally on the same machine running IIS or on a shared folder across the network. This is known usually as the site’s root folder. The root folder contains all kinds of files: web pages, compiled .NET assembles, images, style sheets, JavaScript, and many others. Many of these files are critical to the application and when they change, ASP.NET subsystem needs to pick up these changes automatically to server the latest content to the site visitors. Some of these changes require a site restart for these changes to take effect. These restarts are usually expensive in terms of time needed to server incoming requests. The larger the site and more assemblies it has, the longer it takes to server the first few incoming requests after a restart. Automatic restarts can be controlled through these methods: App pool recycling settings, Asp.NET itself (in “System.Web.dll” assembly that comes with .NET Framework), or programmatically.

By default, ASP.NET adds files watchers to monitor any files’ changes in the root folder and subfolders. Some files/folders are marked as critical and any change in these will cause a site restart. Of these items are the “bin” folder and the “web.config” file. Sometimes, when there are many files changes under the root folder and/or its subfolders, ASP.NET receives an “overwhelming” notification from the files watchers indicating too many files been changed and the watcher(s) can’t report them to ASP.NET individually. This kind of notifications causes ASP.NET to restart the site needlessly. To control this kind of restarts, ASP.NET (as of version .NET 4.5) has a setting to control whether there is a need for monitoring these changes or not and what mode to use for monitoring these changes when enabled. This setting is called “File Change Notifications” (FCN) which is controlled by the “FCNMode” attribute that can be set inside the “<httpRuntime>” element under “<system.web>” section in the “web.confg” file. There are four values for this attribute:

* **NotSet** – will take the setting form an entry in the registry
* **Default** – will install a new file watcher for each critical file in the site
* **Single** – uses a single file watcher for the whole site
* **Disabled** – does not add any watcher for the site files.

When we tested these modes, we found that the two settings: “NotSet” and “Default” have the same behavior. The drawback for this setting is that it creates one file watcher for each single file/folder watched and this can lead to high memory usage, especially when there are many files changes that bubble up through the directories hierarchy. Sometimes, this leads to high memory usage in the site (over 80% of the total server memory). Whenever IIS is setup to restart the site upon high memory usage, this setting causes unwarranted site restarts.

The “Single” mode is generally an improved setting where only one file watcher is used for the whole site. This is better in terms of memory usage, but suffers from a drawback where too many changes occurring within a very short period of time will report an overwhelming notification to ASP.NET which can lead to unjustified site restarts (sometimes many site restarts overlap with each other). To reduce the probability of this happening, the site designers and managers has to try moving out any temporary or file activities (such as image rendering temporary files, very verbose log files, etc.) to a location outside the website. It is notable that some installations doesn’t give the flexibility to being moved outside the root folder, in this case try to minimize the possibility of these change; for example, by limiting the logging level to errors only and not debugging or informational levels in production sites.

The “Disabled” mode means no files watchers are installed at all. This works smoothly for many small sites, but for larger sites this might cause a nuisance as changes to site artifacts (pages, controls, resources, assemblies, etc.) do not auto-restart the site and require manual site restart in order to take effect. It is important to remember here that “web.config” file changes are still picked up and the site will still restart even in this mode.

As of DNN 9.1.1 we added an enhancement to the “Disabled” mode by adding a logic to automatically change the date/time of the “web.config” file after installing modules so that the site restarts automatically in order for the new module(s) to get loaded and take effect. Moreover, for a better diagnostics, we added a single file watcher for the “root” folder (subfolders auto-included) in the site. Whenever any file/folder change occurs in this folder or any subfolder, we log some information (TRACE level only). This will help in debugging unplanned site restarts by showing what file’s activities occur in the site. Note that the default DNN log files are filtered out and no reporting is done for these; otherwise, there will be an endless loop of logging changes of changes of changes… in the log files.

# References:

\* [https://msdn.microsoft.com/en-us/library/system.web.configuration.httpruntimesection.fcnmode(v=vs.110).aspx](https://msdn.microsoft.com/en-us/library/system.web.configuration.httpruntimesection.fcnmode%28v%3Dvs.110%29.aspx)

\* <http://www.dnnsoftware.com/community-blog/cid/154980/aspnet-file-change-notifications-and-dnn>

\* <https://shazwazza.com/post/taming-the-buildmanager-aspnet-temp-files-and-appdomain-restarts/>

\* <https://shazwazza.com/post/all-about-aspnet-file-change-notification-fcn/>

\* <https://blogs.msdn.microsoft.com/tess/2006/08/02/asp-net-case-study-lost-session-variables-and-appdomain-recycles/>