A Guide to Monitoring Folders with EFT

THE INFORMATION IN THIS ARTICLE APPLIES TO:

- EFT v6.0 and later

DISCUSSION

EFT offers a powerful and efficient capability to monitor folders for changes: files being added, removed, or renamed within a folder. In many real-world implementations, Folder Monitors play a major role in both integration with application servers as well as reacting to human processes, where either way data will be deposited in a particular directory somewhere on the network. With that in mind, there may be some question regarding what happens with the context variables and how to properly react to some changes.

Using the Folder Monitor event for scanning files with the Content Integrity Control Action (sending to an ICAP server for scanning) is NOT recommended. Instead, use the File Upload event.

Before we get into the details, remember that each instance of change in a folder is its own discrete event. If two files are added to a directory, each is its own instance and handled independently from the other. If two thousand files are added to a directory, still each file being added is handled on its own.

File Change Triggers

EFT provides three (3) different triggers for reacting to changes to a folder:

1. A file has been added to the directory.
2. A file has been removed from the directory.
3. A file has been renamed within the directory.

By far the most common scenario is to leverage the "if File Change does equal to added" condition. Something new has shown up in the directory, so there's something we should do with it. But how do we know the file's done being written to the directory? And what about the Folder Sweep capability, that option where you can "Scan for files every…" so often? Let's look at that more closely.

"If File Change Does Equal to Added"

EFT is able to know that the file is done being added to the directory when the file handle is closed. A file handle, in this context, means that some application (which could easily be Windows Explorer itself) has taken hold of a file, that it's doing something with it. If what it's doing is writing to a new file, EFT won't then react until that handle is "closed" or released. If this is a new concept, and you'd like to see it in action, there are several utilities freely available from Microsoft's Sysinternals suite, such as Process Monitor that will let you see the continual stream of this otherwise background information.

This concept works very well and reliably when dealing with processes and people on the local LAN. It allows near real-time reaction to files being added to a directory, either on the local server storage or across the network to any CIFS share. Essentially, whether some employee is saving a file in an application to that storage source, moving or otherwise copying and pasting it there, or some application server is doing that kind of work, a new file is created and has new contents written to it for some period of time, and then the handle is closed, indicating that process is finished. It doesn't matter if it takes 5 milliseconds or 5 hours, *EFT won't react until the file handle is closed.*

Scenarios in which to use caution or avoid using Folder Monitor

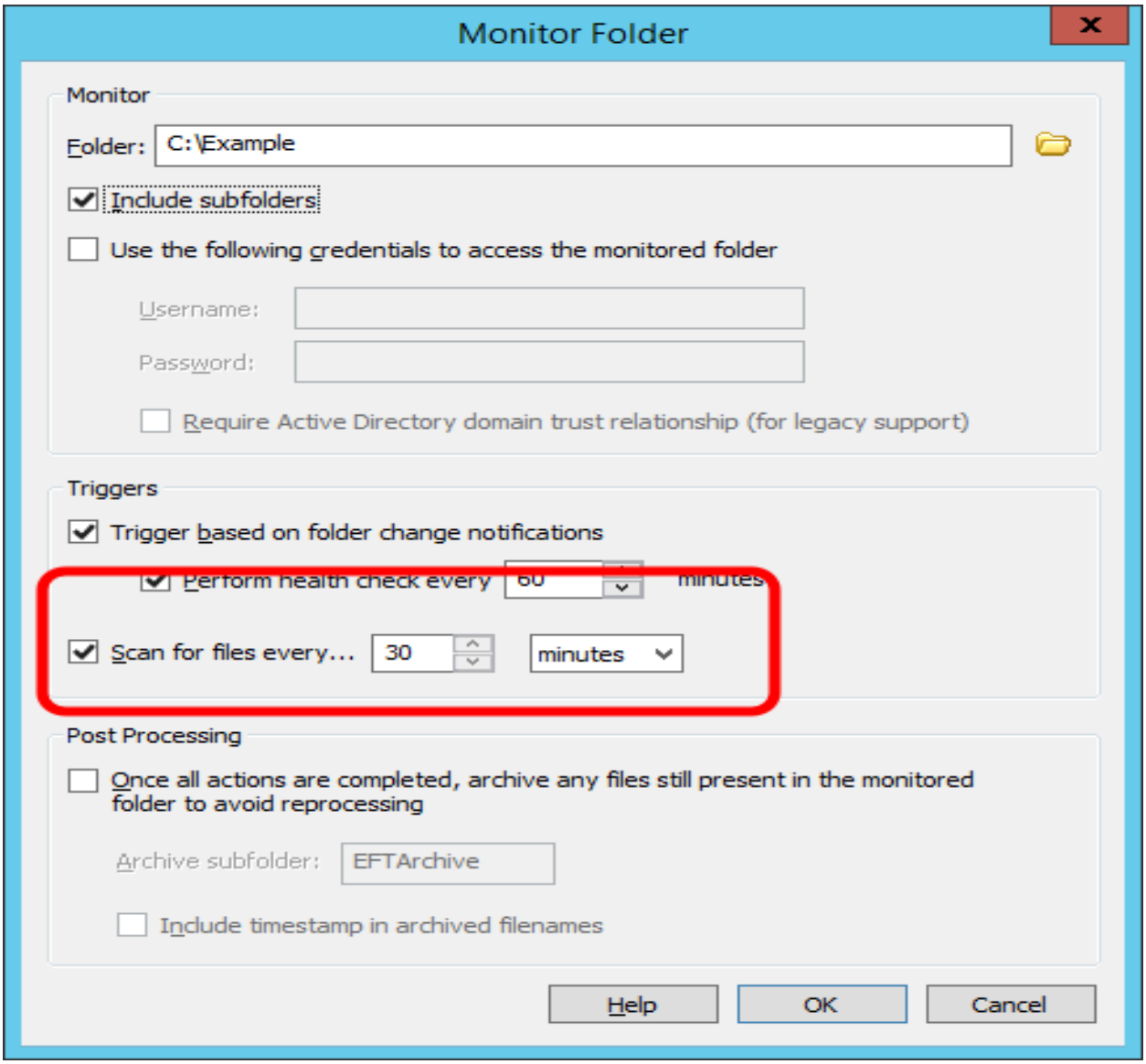
However, there are two basic scenarios to watch out for, scenarios where you must be careful or even avoid using a Folder Monitor altogether. The first scenario is when an application—generally a home-grown or very old legacy application or both—misuses file handles. This is definitely rare but certainly not unheard of. The application may write out a line of a new file, then close the handle, then open a new handle, write out another line, and then close the handle, and so on. This is bad event rule programming and against basic standards. But in this kind of scenario EFT will attempt to grab the file after each new chunk is added since the file handle keeps getting closed. The second scenario is when an EFT administrator tries to use a Folder Monitor to watch where a connected client will be uploading files.

This may work out OK some of the time or even most of the time, but not only is the administrator sacrificing the valuable additional knowledge available with the more appropriate File Uploaded event, but they are also gambling that every future upload will always complete successfully. With the File Uploaded event, EFT only reacts when the file has been fully and completely uploaded successfully. With a Folder Monitor, EFT reacts when the file handle has been closed. If a connected client experiences a simple connection interruption, for example, then the file handle will be closed by EFT, which therefore triggers the Folder Monitor to process a partial file. That's not good. But that's how the administrator

configured EFT, so that's what it's going to do. Software doesn't judge. It simply obeys.

Folder Sweep

When configuring the Folder Monitor properties such as what storage and folder to monitor, you have the option to trigger based on folder change notifications, to scan for files every so often, or both. The folder change notifications are the near real-time pieces of information we can receive from the locally attached storage or a CIFS share, the same information streaming by in the above referenced Sysinternals Process Monitor utility. The Folder Sweep capability, on the other hand, simply looks in the directory for the existence of any file that matches the conditions you've defined in the rule.



So while the folder change notifications are powerful and nearly instantaneous, that won't always work, either because you're monitoring an NFS share (using the Client for NFS feature in Windows Server) or maybe a legacy Samba implementation on an old Linux box that doesn't fully support the CIFS features including folder change notifications.

Additionally, what happens if there's a temporary connection interruption between EFT and the NAS or File Server, and files are added to that folder while EFT can't see that it's happened? The same situation presents itself if the EFT server was down for some period of time, even only shortly and for routine maintenance such as Windows Updates requiring a reboot; files may have been added to the directory while EFT wasn't able to watch for it. Without the Folder Sweep capability to scan for new files, anything added while EFT wasn't able to see it would just sit there.

The Folder Sweep is best used as a safety net, running only a few times a day. The default 30-minute interval is more than sufficient for the vast majority of scenarios. But it can also be used to monitor otherwise blind network storage like NFS and very old Samba implementations. Bear in mind Samba didn't fully support CIFS folder change notifications until version 3.0.25, but since that was released back in early 2007, the hope is that anything in use today would be at least up to that level.

It's important to understand that the Folder Sweep ability only applies to seeing files exist in a directory, implying that it can only be useful when monitoring for new files being added to the directory. EFT enforces this for the benefit of the administrator, explicitly forcing the inclusion of the "if File Change does equal to added" condition in the rule, so that there is no misunderstanding.

"If File Change Does Equal to Removed"

In addition to the assumed default functionality of monitoring for files being added to a directory, many administrators forget or never even realize that EFT can also monitor for files being removed from a directory. This applies to whatever manner causes the removal, whether the obvious deletion of the file, or simply moving the file out of the directory to somewhere else.

This is most commonly used when dealing with "lock" files, where a file may be placed on disk as part of an automated job to indicate that a process is in progress, and the lock file is automatically removed when the process has finished doing its work. That file being removed may then be leveraged by an administrator as the indicator to EFT that it's time to

do some work, triggering the Event Rule.

It's important to bear in mind that many automated functions of EFT itself may cause files to be removed from a directory. Encrypting or decrypting a file with the OpenPGP action, for example, will cause the original file to be removed from the directory and a new file (the result of the encryption or decryption action) to be added to the directory. Using the Move action or the "Perform file Delete operation" will of course also trigger this condition. Take care not to cause some form of recursion. It's not common to see, but it has happens, especially to administrators who are part of a large team and may not be aware of what some other administrator may be doing in that directory already.

"If File Change Does Equal to Renamed"

Similarly to files being removed, reaction to a file being renamed in a directory is almost always associated with flag files or lock files. For example, a file named PROCESSING may be renamed to READY. Or a file called job.lock may be renamed to job.ready. While this is a fairly primitive integration method, it has been used successfully for many years when automated systems have no better way to talk to each other.

The one important thing to remember with reacting to files being renamed is that suddenly an additional set of File System context variables come into play. The typical %FS.FILE_NAME% and %FS.PATH% values, for example, refer to the original file that was renamed. To deal with the "destination" file, the results of the rename, you would instead use the variables that have DST prepended to their name, such as %FS.DST_FILE_NAME% and %FS.DST_PATH%. For example, if you trigger on a rename and then attempt to move %FS.PATH% to a different location, it will fail, because that file that was renamed no longer exists. Instead, you would need to move %FS.DST_PATH% since that represents the path of the file that resulted from the rename occurring.

GlobalSCAPE Knowledge Base
https://kb.globalscape.com/Knowledgebase/11243/A-Guide-to-Monitoring-Folder...