

# Scripting

Scripting examples and techniques in SyncJS, the JavaScript-based scripting language behind Syncplify AFT!

- [How to monitor a local folder and upload files to your SFTP server when they change](#)

# How to monitor a local folder and upload files to your SFTP server when they change

Let's say you have an SFTP server somewhere, and you want to use it as some form of "real-time backup". That implies monitoring a local folder/directory on your computer's hard disk (or SSD), and:

- detect when new files are created and upload them
- detect when files are modified and upload them

Furthermore, there are several other aspects to consider. For example:

- your SFTP server may (should) not allow your client to be always connected
- you may need to "delay" your uploads, because the OS (file-system) needs time to finish writing the local file before you can access it and upload it to the remote SFTP server

AFT! can help you with all of that.

Here below you can see a well-commented AFT! script (in pure SyncJS language) that shows you how to do all of the above the right way, taking all the above-mentioned circumstances into account:

```
{
  // Let's enable console feedback, in case we're running this script via the shell
  ConsoleFeedback = true;

  // First, let's create the file-system watcher
  watcher = new FsWatcher();

  // Then we elect to delay notification by *at least* 300 seconds (5 minutes)
  // (useful to allow the file system to finish whatever operation is ongoing)
  watcher.DelayBySeconds = 300;

  // We may choose *not* to be notified of certain events
  watcher.NotifyRename = false;
  watcher.NotifyRemove = false;
  watcher.NotifyChmod = false;
```

```

// We can specify inclusion and exclusion filters (optional, not mandatory)
watcher.InclusionFilter = ['*.*'];
watcher.ExclusionFilter = ['notes.txt', 'budget.xlsx'];
// Almost ready, let's tell the object what folder we want to monitor
watcher.WatchDir('C:\\TestFolder', false);
// And then start the watcher
watcher.Start();

// We need to keep checking events indefinitely, an endless cycle is what we need
while (true) {
    // Let's sleep for 500 milliseconds at each cycle, to keep CPU usage low
    Sleep(500);
    // When inside an endless cycle, it's always safe to check if we received a Halt signal at every cycle
    if (HaltSignalReceived()) {
        break;
    }
    // No Halt signal? Good, then let's acquire the list of pending event that we need to process
    evt = watcher.Events()
    // Do we have at least 1 event to process?
    if (evt.length > 0) {
        // We only connect to the server IF there are events to be processes
        var scli = new SftpClient();
        scli.Host = 'your.sftpserver.com:22';
        scli.User = 'your_username';
        scli.Pass = 'your_password';
        scli.Options.UploadPolicy = AlwaysOverwrite;
        if (scli.Connect()) {
            // Cycle over all pending events...
            for (var i = 0; i < evt.length; i++) {
                if (evt[i].Event == 'WRITE') {
                    // If it is a WRITE event (new or modified file) let's upload it to the server
                    scli.UploadWithPath(evt[i].Object, '/destinationpath', 1);
                }
            }
            // Do not forget to close the connection
            scli.Close();
        }
        // Set the client object to null to save memory
        scli = null;
    }
}

```

```
}  
}
```

Happy coding!