Breaking Changes: AFT! v3 to v4

This document lists every change in AFT! v4 that requires a script or configuration to be updated before it will run as expected. Read it before upgrading any production instance.

The short answer for most users: **the vast majority of AFT! v3 scripts will run in v4** without any modification. The only scripts that need changes are those that use any of the features listed below.

`FsWatcher.Start()` now requires a callback

V3 pattern (no longer works):

```
watcher.Start();


while (true) {
    Sleep(500);
    if (HaltSignalReceived) {
        break;
    }
    var evt = watcher.Events();
    for (var i = 0; i < evt.length; i++) {
        if (evt[i].Event == 'WRITE') {
            doSomethingWith(evt[i].Object);
        }
    }
}
```

V4 pattern (new):

```
watcher.Start(function(evt) {
    // evt = { TimeStamp, Event, Object }
    if (evt.Event == EVT_WRITE) {
        doSomethingWith(evt.Object);
    }
```

```
});


WaitForHaltSignal();
watcher.Stop();
```

What changed:

- `Start()` now requires a callback function as its only argument. Calling it with no arguments silently does nothing.
- `Events()` has been removed. There is no longer a queue to poll.
- `WaitForHaltSignal()` replaces the `while/Sleep/HaltSignalReceived` loop. It blocks until a halt signal is received, using no CPU and producing no log noise.
- `Stop()` is a new required step after `WaitForHaltSignal()` returns. It waits until the internal event goroutine has fully exited before returning, ensuring no further callbacks fire after the script continues past that point.

Event-type note: in v3 the event types were uppercase strings (`"WRITE"`, `"CREATE"`, `"REMOVE"`, `"RENAME"`, `"CHMOD"`). In v4 they are now named constants (`EVT_WRITE`, `EVT_CREATE`, `EVT_REMOVE`, `EVT_RENAME`, `EVT_CHMOD`). Update any `==` comparisons accordingly.

RemoteWatcher constructor now takes a VFS name

V3 pattern (no longer works):

```
var rcli = new SftpClient();
rcli.Host  = 'files.example.com:22';
rcli.User  = 'uploader';
rcli.Pass  = 's3cr3t';
rcli.Connect();


var rw = new RemoteWatcher(rcli);
```

V4 pattern (new):

```
// "production-sftp" is the name of an entry in the AFT! VFS Library
var rw = new RemoteWatcher("production-sftp");
```

What changed: the constructor no longer accepts a live client object. It accepts a string: the name of a named VFS connection profile stored in the AFT! Virtual File Systems library. The engine resolves the profile, decrypts its credentials, and opens the connection internally. Credentials never appear in script source code in v4.

To migrate: create a Virtual File System entry for each remote system your RemoteWatcher scripts target, then replace the client construction code with `new RemoteWatcher("profile-name")`.

`HaltSignalReceived` is now a function, not a property

V3 pattern (no longer works):

```
if (HaltSignalReceived) {   // property access, no parentheses
    break;
}
```

V4 pattern (new):

```
if (HaltSignalReceived()) {   // function call, parentheses required
    break;
}
```

Why this matters: in v4, `HaltSignalReceived` is a JavaScript function. A bare reference to a function (without calling it) is always truthy in JavaScript, regardless of the actual halt state. A script that uses `if (HaltSignalReceived)` without parentheses will evaluate the condition as `true` on the very first iteration and **exit immediately**, never processing any events.

Note: most scripts that reach v4 with `FsWatcher` or `RemoteWatcher` will be rewritten to use the callback pattern and `WaitForHaltSignal()` anyway, at which point this property is rarely needed. It remains available for scripts that prefer explicit polling.

`*FromSecret` credential properties have been removed

In v3, every client object exposed shortcut properties that accepted a secret name and resolved it internally. These properties no longer exist in v4.

V3 pattern (no longer works):

```
var sftpcli = new SftpClient();
sftpcli.User           = 'uploader';
sftpcli.PassFromSecret = 'name-of-my-secret';       // v3 shortcut property


var s3cli = new S3Client();
s3cli.APIKeySecretFromSecret = 'aws-secret-name';   // v3 S3 shortcut property


cli.Options.OTFEKeyFromSecret = 'otfe-key-secret';  // v3 OTFE shortcut property
```

V4 pattern (new):

```
var sftpcli = new SftpClient();
sftpcli.User = 'uploader';
```

```
sftpcli.Pass = GetSecret('name-of-my-secret');      // call GetSecret() inline


var s3cli = new S3Client();
s3cli.AccessSecret = GetSecret('aws-secret-name');  // AccessSecret is the v4 field name


cli.Options.OTFEKey = GetSecret('otfe-key-secret'); // OTFEKey is the v4 field name
```

What changed: the `*FromSecret` shortcut properties (`PassFromSecret`, `APIKeySecretFromSecret`, `Options.OTFEKeyFromSecret`, and any others of the same pattern) have been removed. In v4 `GetSecret("name")` is a first-class global function. Call it directly and assign its return value to the appropriate credential field.

Migration: search all script files for the string `FromSecret`. Each occurrence maps mechanically to a `GetSecret()` call: replace `client.XFromSecret = "name"` with `client.X = GetSecret("name")`, substituting the base field name (`Pass`, `AccessSecret`, `OTFEKey`, etc.).

Admin accounts are not migrated

AFT! v3 stored admin passwords as SHA-256 hashes of `random_salt + username + password`. AFT! v4 uses **bcrypt**, which is an incompatible format. Because the plain-text passwords cannot be recovered from the v3 hashes, migrating admin accounts would only produce accounts that nobody could log into.

`aft import-from-aft3`, therefore, does not import any admin profiles. AFT! v4 creates the initial admin account during first-run setup. Any additional admin accounts must be recreated manually in the v4 web UI after the initial setup is complete.

Blockly script type removed

AFT! v3 allowed scripts to be authored in Blockly (a visual block-based editor). AFT! v4 supports only SyncJS JavaScript scripts. Blockly scripts are skipped during migration with a warning listing each one by name; they cannot be run in v4.

If any Blockly scripts are still in use, their logic must be rewritten from scratch in JavaScript after upgrading.

Revision #1
Created 12 April 2026 21:35:33 by DevTeam
Updated 12 April 2026 22:00:30 by DevTeam