FreeFileSync User Manual:

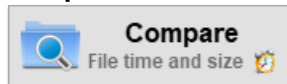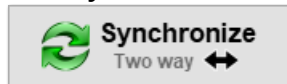— Folder Comparison and Synchronization —

Basic Usage:

1. Choose left and right folders.

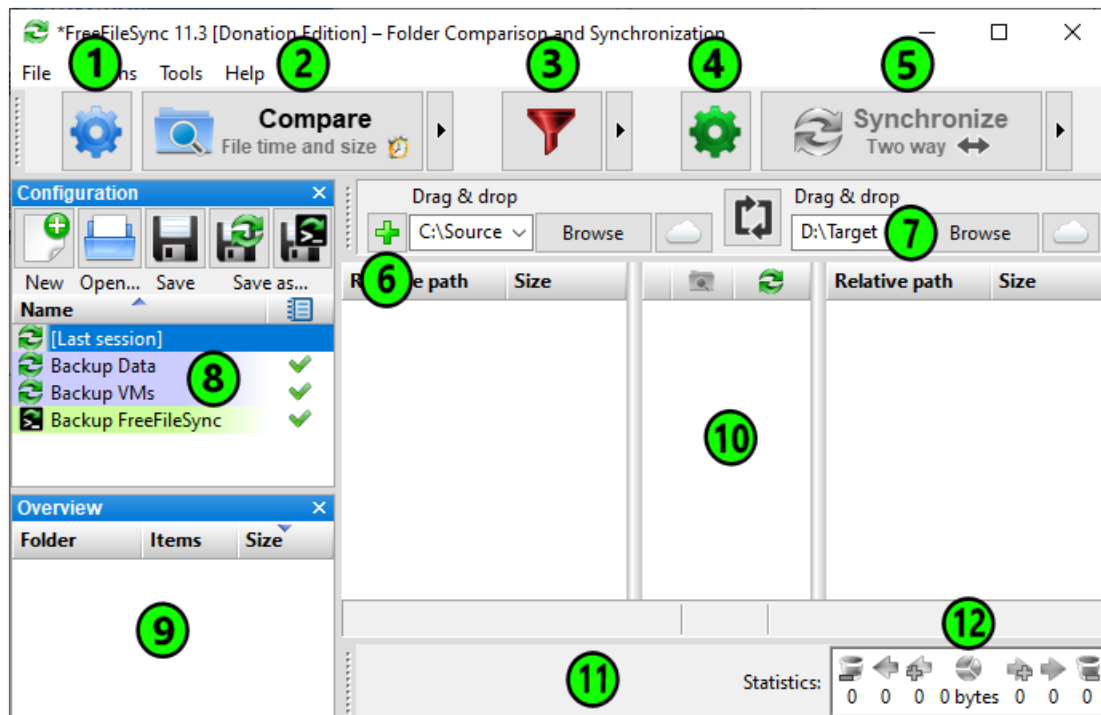2. **Compare** them.

3. Select synchronization settings.

4. Press **Synchronize** to begin synchronization.

> **Note**
> For more detailed explanations on how to set up the most common synchronization scenarios,
> have a look at the **FreeFileSync video tutorials**.

Main Dialog Overview

1. Change comparison settings
2. Start comparison
3. Include/exclude specific files
4. Change synchronization settings
5. Start synchronization
6. Add folder pairs
7. Select left and right folders
8. Save/load configuration
9. Tree overview panel
10. Synchronization preview
11. Select categories to show on grid
12. Synchronization statistics

Command Line Usage

FreeFileSync supports additional synchronization scenarios via a command line interface. To get a syntax overview, open the console, go to the directory where FreeFileSync is installed and type:

```
FreeFileSync -h      or      FreeFileSync --help
```
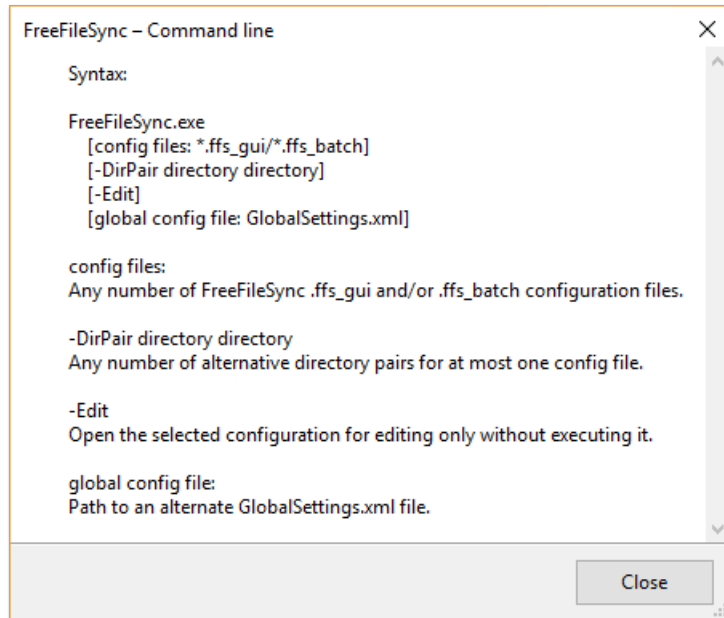

1. Run a FreeFileSync batch job

In order to start synchronization in batch mode, supply the path of a ffs_batch configuration file as the first argument after the FreeFileSync executable:

```
FreeFileSync "D:\Backup Projects.ffs_batch"
```

After synchronization one of the following status codes is returned:

Exit Codes
0 Synchronization completed successfully
1 Synchronization completed with warnings
2 Synchronization completed with errors
3 Synchronization was aborted

You can evaluate these codes from a script (e.g. a cmd or bat file on Windows) and check if synchronization completed successfully:

```
"C:\Program Files\FreeFileSync\FreeFileSync.exe" "D:\Backup
Projects.ffs_batch"
if not %errorlevel% == 0 (
  ::if return code is 1 or greater, something went wrong, add
special treatment here
  echo Errors occurred during synchronization...
  pause & exit 1
)
```

> **Attention**
>
> If you are running the batch job unattended, make sure your script is not blocked showing a notification dialog. Consider the following options when setting up the FreeFileSync batch job:
>
> - Enable **Auto-Close** to skip the summary dialog after synchronization.
>
> - Set up error handling to **Ignore errors** or **Cancel** to stop the synchronization at the first error.

2. Start a FreeFileSync GUI configuration

If you pass a ffs_gui file, FreeFileSync will start in GUI mode and immediately start comparison (but only if all directories exist):

```
FreeFileSync "D:\Manual Backup.ffs_gui"
```

3. Customize an existing configuration

You can replace the directories of a given ffs_gui or ffs_batch configuration file by using the `DirPair` parameter:

```
FreeFileSync "D:\Manual Backup.ffs_gui" -dirpair C:\NewSource
D:\NewTarget
```

4. Merge multiple configurations

When more than one configuration file is provided, FreeFileSync will merge everything into a single configuration with multiple folder pairs and start in GUI mode:

```
FreeFileSync "D:\Manual Backup.ffs_gui" "D:\Backup
Projects.ffs_batch"
```

5. Use a different GlobalSettings.xml file

By default, FreeFileSync uses a single GlobalSettings.xml file containing options that apply to all synchronization tasks; for examples see **Expert Settings**. If you want FreeFileSync to use a different settings file instead, just add the path via command line:

```
FreeFileSync "D:\Different GlobalSettings.xml"
```

Comparison Settings
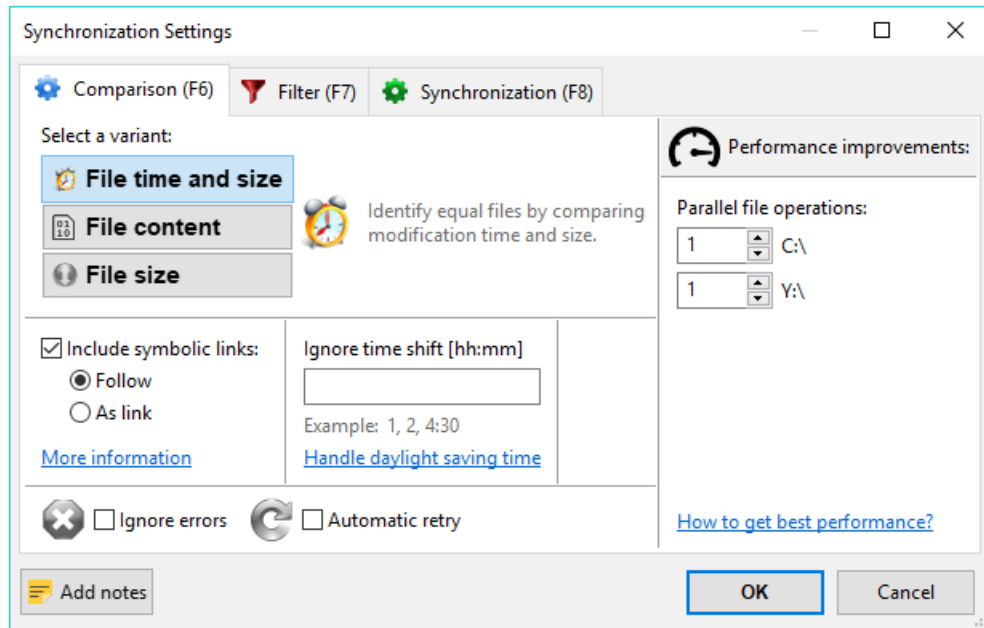
Comparison Variants

When comparing two folders, FreeFileSync analyses the**relative paths** of the contained files. If the relative path matches, FreeFileSync decides how the file pair is categorized by considering the selected comparison variant:

1. Compare by *File time and size*

This variant considers two files equal when both **modification time and file size** match. It should be selected when synchronizing files with a backup location. Whenever a file is changed, its file modification time is also updated. Therefore, a comparison by *File Time and size* will detect all files that should be synchronized. The following categories are distinguished:

I. **file exists on one side only**
   - left only
   - right only
II. **file exists on both sides**
   i. **different date**
      - left newer
      - right newer
   ii. **same date**
      - equal
      - conflict (same date, different size)

2. Compare by *File content*

Two files are marked as equal if they have **identical content**. This variant should be selected when doing consistency checks to see if the files on both sides are bit-wise identical. Naturally, it is the slowest of all comparison variants, so its usefulness for the purpose of synchronization is limited. If used for synchronization, it can serve as a fallback when modification times are not reliable. For example certain mobile phones and legacy FTP servers do not preserve modification times, so the only way to detect different files when the file sizes are the same is by reading their content.

I. **file exists on one side only**

- left only
- right only

II. **file exists on both sides**
- equal
- different content

3. Compare by *File size*

Two files are considered equal if they have the **same file size**. Since it's possible for files that have the same size to have different content, this variant should only be used when file modification times are not available or reliable, e.g. in certain MTP and FTP synchronization scenarios, and where a comparison by content would be too slow.

I. **file exists on one side only**
- left only
- right only

II. **file exists on both sides**
- equal
- different size

Symbolic Link Handling

FreeFileSync lets you choose to include symbolic links (also called symlinks or soft links) when scanning directories rather than skipping over them. When included, you can select between two ways to handle them:

1. **Follow:** Treat symbolic links like the object they are pointing to. Links pointing to directories are traversed like ordinary directories and the target of each link is copied during synchronization.

2. **As link:** Include the symbolic link object directly. Symbolic links will be shown as separate entities. Links pointing to directories are not traversed and the link object itself is copied during synchronization.

> **Note**
> - FreeFileSync considers the following items as "symbolic links":
>   – file system symbolic links
>   – volume mount points (NTFS)
>   – junction points (NTFS)
>   – WSL symlinks
>   – Google Drive shortcuts
>
> - Windows: Copying symbolic links requires that FreeFileSync is started with administrator rights.

Daylight Saving Time (Windows)

A common problem synchronization software has to handle is +-1 hour file time shifts after a Daylight Saving Time (DST) switch has occurred. This can be observed, for example, when a FAT32- or exFAT-formatted volume (in the following called "FAT") is compared against an NTFS volume, like when synchronizing a USB memory stick against a local disk. Files that previously appeared to be in sync are now shown with a one hour modification time offset, although they have not been modified by the user or the operating system.

The reason for this behavior lies in the way NTFS and FAT store file times: NTFS stores time in UTC format, while FAT uses local time.

When times of these two different formats are compared, one format has to be converted into the other first. In either way, Windows uses the current DST status as well as the current time zone for its calculations. Consequently, the result of this comparison is dependent from current system settings with the effect that file times that used to be the same show up as different after a DST switch or when the time zone has changed.

For a detailed discussion about this issue see:
[↗ **Beating the Daylight Savings Time Bug and Getting Correct File Modification Times**](#)

Solutions:

1. In FreeFileSync's comparison settings you can enter one or more time shifts to ignore during comparison: If you need to handle differences due to daylight saving time, enter a single one hour shift. If the differences are caused by changing the time zone, enter one or more time shifts as needed.
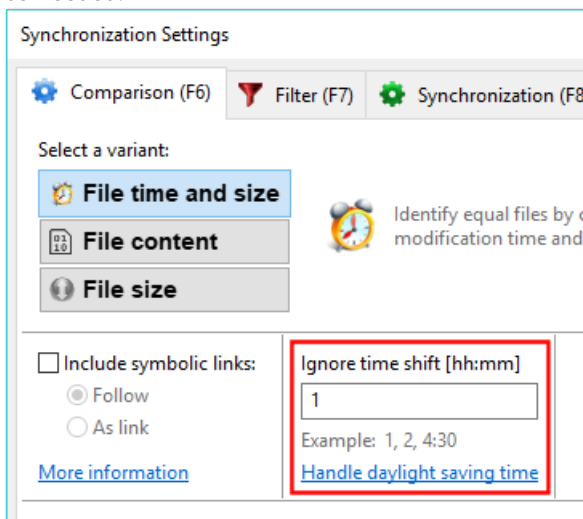

> **Note**
> File times have to be equal or differ by exactly the time shift entered to be considered the same. Therefore, the time shift setting should not be confused with a time interval or tolerance.

2. Alternatively, you can avoid the problem in the first place by only synchronizing from FAT to FAT or NTFS to NTFS file systemsSince most local disks are formatted with NTFS and USB memory sticks with FAT, this situation could be handled by formatting the USB stick with NTFS as well.

Exclude Files and Folders via Filter
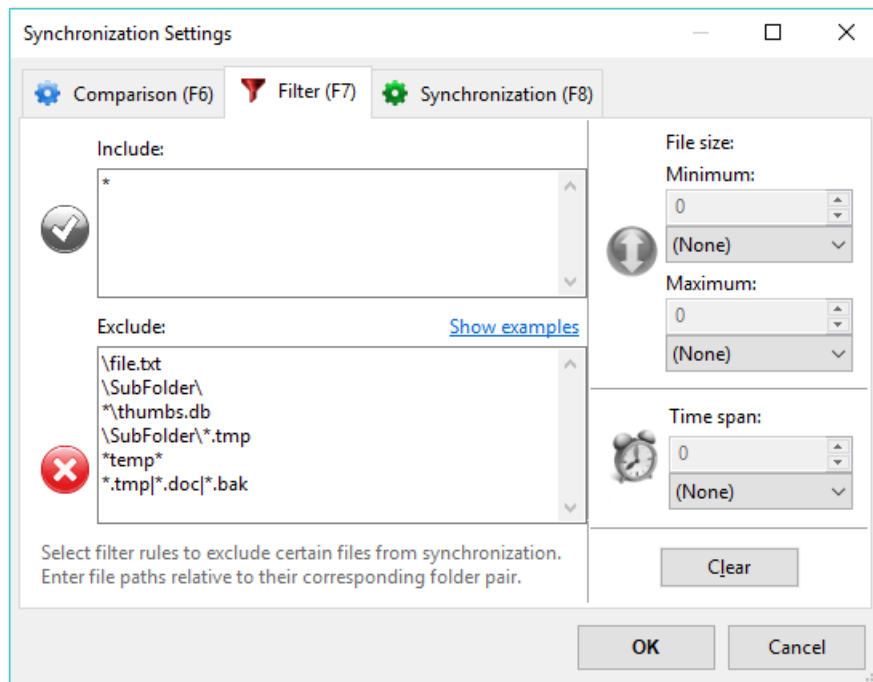
Files and folders are only considered for synchronization if they pass all filter rules: They have to match **at least one** entry in the include list and **none** of the entries in the exclude list as presented in the filter configuration dialog.

- Each filter item must be a file or folder path **relative** to the selected folder pair.
- Multiple items must be separated by **|** or a **new line**.
- Wild cards may be used: ***** means **zero or more** characters
  ? represents **exactly one** character
  ?* matches **one or more** characters
- Filter matching is **case-insensitive**

Example: Match items of a folder pair

The following filter phrases assume a folder pair `C:\Source <—> D:\Target` *and can be used for the include as well as exclude filter.*

Description	Filter phrase
Single file (or folder) `C:\Source\`**`file.txt`**	`\file.txt`
Single folder `C:\Source\`**`SubFolder`**	`\SubFolder\`
All files (and folders) named **`thumbs.db`**	`*\thumbs.db`
All files (and folders) starting with the **Z** character	`*\z*`
All **`*.tmp`** files (and folders) in `C:\Source\`**`SubFolder`**	`\SubFolder\*.tmp`
Files and folders containing **`temp`** in their path	`*temp*`
Multiple entries (separated by vertical bar)	`*.tmp | *.doc | *.bak`
All **sub**folders of the folder pair	`*\`
Files/folders **inside sub**folders of the folder pair	`?*\?*`
All files (**but not folders**)	`*:`

Example: Complex filter rules with exceptions

Complex filter requirements can often be solved by using **two folder pairs** with the same source and target paths but **different local filter**. The first folder pair handles the default case. The second folder pair the exception.

*Exclude a sub folder except for *.txt files by using two folder pairs:*

```
C:\Source <—> D:\Target    local exclude filter: \SubFolder\
C:\Source <—> D:\Target    local include filter: \SubFolder\*.txt
```

Example: Exclude empty folders

Set **`*:`** as include filter to match all files, but not folders. During synchronization some excluded folders will still be created if needed, but only if they contain at least one non-excluded item, that is, when they are not empty.

> **Note**
> - For simple exclusions: Instead of typing the filter phrase manually, go the the FreeFileSync main window, right-click one or more files from the list, and exclude via the **context menu**.
>
> - A filter phrase can match both **file and folder** paths by default. To match only one of them, you can give a hint:
>   1. Files only: append a colon (**:**)
>   2. Folders only: append a path separator (**/** or ****)
>
> - If the filter is matching a folder, all its files and subfolders are also **(implicitly) matched**. Thus the filter phrases `SubFolder\` and `SubFolder\*` are synonymous.
>
> - Both slash (**/**) and backslash (****) can be used as the path separator character.

Expert Settings

FreeFileSync has a number of special-purpose settings that can only be accessed by manually opening the global configuration file `GlobalSettings.xml`. Note that this file is read once when FreeFileSync starts and saved again on exit. Therefore, you should **apply manual changes only while FreeFileSync is not running.** For the portable FreeFileSync variant the file is found in the installation folder, for local installations go to:

Windows: **%AppData%\FreeFileSync**

Linux: **~/.config/FreeFileSync**

macOS: **~/Library/Application Support/FreeFileSync**

```
<?xml version="1.0" encoding="UTF-8"?>
<FreeFileSync XmlType="GLOBAL">
    <General>
        <FileTimeTolerance Seconds="2"/>
        <RunWithBackgroundPriority Enabled="false"/>
        <LockDirectoriesDuringSync Enabled="true"/>
        <VerifyCopiedFiles Enabled="false"/>
```

Contents of GlobalSettings.xml

FileTimeTolerance:
By default file modification times are allowed to have a 2 second difference while still being considered equal. This is required by FAT/FAT32 file systems which store file times only with a 2-second precision.

RunWithBackgroundPriority:
While synchronization is running, other applications that are accessing the same data locations may experience a noticeable slowdown. Enable this setting to lower FreeFileSync's file access priority at the cost of a *slower* synchronization speed.

LockDirectoriesDuringSync:
In order to prevent multiple synchronization tasks from reading and writing the same files, FreeFileSync instances are serialized with lock files (`sync.ffs_lock`). The lock files are only recognized by FreeFileSync and make sure that at most, a single synchronization is running against a certain folder at a time while other instances are queued to wait. This ensures that only consistent sets of files are subject to synchronization. The primary use case are network synchronization scenarios where multiple users run FreeFileSync concurrently against a shared network folder.

VerifyCopiedFiles:
If active, FreeFileSync will binary-compare source and target files after copying and report verification errors. Note that this may double file copy times and is no guarantee that data has not already been corrupted prior to copying. Additionally, corruption may be hidden by deceptively reading valid data from various buffers in the application and hardware stack:
[↗ **Does the CopyFile function verify that the data reached its final destination successfully?**](#)

External Applications

When you double-click on one of the rows on the main dialog, FreeFileSync opens the operating system's file browser by default:

```
Windows:  explorer.exe /select, "%local_path%" & exit 0
macOS:    open -R "%local_path%"
Linux:    xdg-open "$(dirname "%local_path%")"
```

To customize this behavior, or integrate other external applications into FreeFileSync, navigate to **Menu → Tools → Options → Customize context menu** and add or replace a command.

All entries can be accessed quickly by pressing the associated **numeric keys 0–9** or via the context menu that is shown after a right mouse click. The **first entry** can also be executed by **double-clicking** on an item.

In addition to regular **Macros**, the following special macros are available:

Macro	Description
`%item_path%`	Full file or folder path
`%local_path%`	Creates a temporary local copy for files located on SFTP and MTP storage. Otherwise identical to %item_path% for local files and network shares.
`%item_name%`	File or folder name
`%parent_path%`	Parent folder path

- To refer to the item on the opposite side, append "**2**" to the macro name:
  `%item_path`**2**`%, %local_path`**2**`%, %item_name`**2**`%, %parent_path`**2**`%.`

- To generate a list including all selected items (separated by space), append "**s**" to the macro name:
  `%item_path`**s**`%, %local_path`**s**`%, %item_name`**s**`%, %parent_path`**s**`%.`

Examples:

- Start a file content comparison tool:

  Windows: ↗WinMerge
  ```
  "C:\Program Files (x86)\WinMerge\WinMergeU.exe" "%local_path%"
  "%local_path2%"
  ```

  macOS: opendiff (requires Xcode)
  ```
  opendiff "%local_path%" "%local_path2%"
  ```

  Ubuntu: kompare (`sudo apt install kompare`)
  ```
  kompare "%local_path%" "%local_path2%"
  ```

- Show file in Windows Explorer:
  ```
  explorer.exe /select, "%local_path%" & exit 0
  ```

> **Note**
> Explorer.exe does not set an exit code, but FreeFileSync will show an error message if it does not find **exit code = 0** ("Success"). To mitigate, append (**&**) command **exit 0** to set the exit code explicitly.

- Open command prompt for the selected item:
```
start cmd.exe /k cd /D "%parent_path%"
```
> **Note**
> FreeFileSync hides the console window, so **start** opens a new window. **cmd.exe /k** runs the following command without immediately exiting the console. **cd** navigates to the directory, even if it's on a different volume (**/D**).

- Copy item path to Clipboard (as alternative to CTRL + C)
```
echo %item_path%| clip
```

- Write list of selected file paths to a text file:
```
echo %item_path% >> %csidl_Desktop%\file_list.txt
```

- Preview files using Quick Look on macOS:
```
qlmanage -p "%local_path%"
```

- Pass a list of selected files to a script as command line arguments:
```
C:\my-script.cmd "%local_paths%"
```

> **Note**
> Macros need to be protected with quotation marks if they resolve to file paths that could contain whitespace characters.

Macros

All directory paths may contain macros that are expanded during synchronization. The beginnings and ends of each macro are marked by a **%** character. In addition to special macros handling time and date, the **operating system's environment variables** may also be used.

Internal Macros

Macro	Example	Format
%Date%	2025-03-27	[YYYY-MM-DD]
%Time%	090342	[hhmmss]
%TimeStamp%	2025-03-27 090342	[YYYY-MM-DD hhmmss]
%Year%	2025	
%Month%	03	[01–12]
%MonthName%	Mar	[Jan–Dec]
%Day%	27	[01–31]
%Hour%	09	[00–23]
%Min%	03	[00–59]
%Sec%	42	[00–59]
%WeekDay%	4	[1–7] *week begin may vary locally*
%WeekDayName%	Thu	[Mon–Sun]
%Week%	13	[01–52] *calendar week*

Environment Variables (Windows)

Macro	Example
%AllUsersProfile%	C:\ProgramData
%AppData%	C:\Users\Zenju\AppData\Roaming
%ComputerName%	Zenju-PC
%LocalAppData%	C:\Users\Zenju\AppData\Local
%ProgramData%	C:\ProgramData
%ProgramFiles%	C:\Program Files
%ProgramFiles(x86)%	C:\Program Files (x86)
%Public%	C:\Users\Public
%Temp%	C:\Windows\Temp
%UserName%	Zenju
%UserProfile%	C:\Users\Zenju
%WinDir%	C:\Windows

Special Folder Locations (Windows)

Macro	Example
`%csidl_Desktop%`	`C:\Users\Zenju\Desktop`
`%csidl_Documents%`	`C:\Users\Zenju\Documents`
`%csidl_Pictures%`	`C:\Users\Zenju\Pictures`
`%csidl_Music%`	`C:\Users\Zenju\Music`
`%csidl_Videos%`	`C:\Users\Zenju\Videos`
`%csidl_Downloads%`	`C:\Users\Zenju\Downloads`
`%csidl_Favorites%`	`C:\Users\Zenju\Favorites`
`%csidl_Resources%`	`C:\Windows\Resources`
`%csidl_QuickLaunch%`	`C:\Users\Zenju\AppData\Roaming\Microsoft\Internet Explorer\Quick Launch`
`%csidl_StartMenu%`	`C:\Users\Zenju\AppData\Roaming\Microsoft\Windows\Start Menu`
`%csidl_Programs%`	`C:\Users\Zenju\AppData\Roaming\Microsoft\Windows\Start Menu\Programs`
`%csidl_Startup%`	`C:\Users\Zenju\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\StartUp`
`%csidl_Nethood%`	`C:\Users\Zenju\AppData\Roaming\Microsoft\Windows\Networ Shortcuts`
`%csidl_Templates%`	`C:\Users\Zenju\AppData\Roaming\Microsoft\Windows\Templa`

Note: Most of the macros above have a variant for public folders, e.g. `%csidl_Documents%` has `%csidl_PublicDocuments%`.

Hint: You can add flexibility to an ffs_batch configuration file by creating new temporary environment variables in a bat or cmd file that are evaluated by FreeFileSync at runtime:

Example:

The FreeFileSync batch file `C:\SyncJob.ffs_batch` contains macro `%MyVar%` instead of an absolute target folder and is invoked by a cmd file:

```
set MyVar=C:\Target
"C:\Program files\FreeFileSync\FreeFileSync.exe"
C:\SyncJob.ffs_batch
::%MyVar% is resolved as C:\Target during synchronization
```

Note
Temporary environment variables created with the `set` command are only valid if the synchronization is started by calling the FreeFileSync executable directly. Using `start /wait` would create a new program context without these temporary variables.

Performance Improvements

FreeFileSync can be set up to issue multiple file accesses in parallel. This speeds up synchronization times dramatically in cases where single I/O operations have significant latency (e.g. long response times on a slow network connection) or they cannot use the full bandwidth available (e.g. an FTP server enforcing a speed limit for each connection).
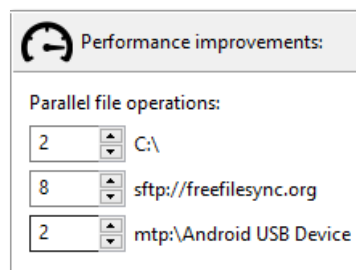
The number of parallel file operations that FreeFileSync should use can be set up for each device individually in the **Comparison Settings** dialog. It is evaluated for all folder pairs of a configuration as follows:

- **During comparison** FreeFileSync groups all folders by their root devices.

  For example, consider a configuration with two folder pairs and parallel file operations set up:

			Device root	Parallel operations
C:\Source	↔	D:\Target	C:\	1
C:\Source2	↔	E:\Target	D:\	2
			E:\	3

  FreeFileSync will put the folders C:\Source and C:\Source2 into the same group and allow only 1 file operation at a time. Folder D:\Target will be traversed using 2 operations, and E:\Target using 3 operations at a time. In total FreeFileSync will be scanning all four folders employing 6 file operations in parallel.

- **When synchronizing** a folder pair FreeFileSync will use the **maximum** of the number of parallel operations that the two folders support.

  In the previous example the folder pair C:\Source ↔ D:\Target will be synchronized using 2 parallel operations, and C:\Source2 ↔ E:\target will be using 3.

> **Note**
> FreeFileSync implements parallel file operations by opening multiple connections to a device. Some devices like SFTP servers have limits on how many connections they allow and will fail if too many are attempted; see **(S)FTP Setup**.

— Automated Synchronization —
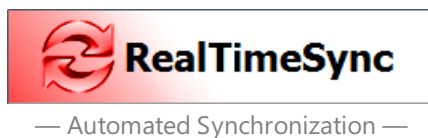
The function of RealTimeSync is to **execute a command line each time it detects changes** in one of the monitored directories, or when a directory becomes available (e. g. insert of a USB-stick). Usually this command line will **trigger a FreeFileSync batch job**.

RealTimeSync receives change notifications directly from the operating system in order to avoid the overhead of repeatedly polling for changes. Each time a file or folder is created/updated/deleted in the monitored directories or their sub directories, RealTimeSync **waits until a user-configurable idle time has passed** in which no further changes were detected, and then runs the command line. This makes sure the monitored folders are not in heavy use when starting a synchronization.

Example: Real time synchronization using FreeFileSync

Start RealTimeSync.exe located in FreeFileSync's installation directory and enter the folders you want to monitor. Instead of doing this manually you can import an ffs_batch file via **Menu → File → Open** or simply via **drag and drop**. RealTimeSync will not only extract all directories relevant for synchronization, but will also set up the command line to execute the ffs_batch file each time changes are detected. Now press **Start** to begin monitoring.

> **Note**
> - The command should **not block** progress. If you call a FreeFileSync batch job, make sure it won't show any popup dialogs. See notes in **Command Line Usage**.
> - RealTimeSync will **skip showing the main dialog** and begin monitoring immediately if y o u **pass an ffs_real** configuration file as the first command line argument to RealTimeSync.exe. This can be used to integrate RealTimeSync into the operating system's auto start:
>
>   ```
>   "C:\Program Files\FreeFileSync\RealTimeSync.exe" "D:\Backup
>   Projects.ffs_real"
>   ```
>
> - You can also **pass an ffs_batch** file as first argument, which RealTimeSync will automatically convert into a ffs_real configuration with default settings (e.g. 10 seconds idle time).
> - RealTimeSync does **not require you to start FreeFileSync**. It can also be used in other scenarios, like sending an email whenever a certain directory is modified.

Example: Automatic synchronization when a USB stick is inserted

Save an ffs_batch configuration in the USB stick's root directory, e.g. H:\ and let FreeFileSync run it when the stick is mounted. But, instead of hard coding the USB drive letter H:\ (which may change occasionally), refer to the USB stick via its **volume name** instead.

Configure RealTimeSync as follows:
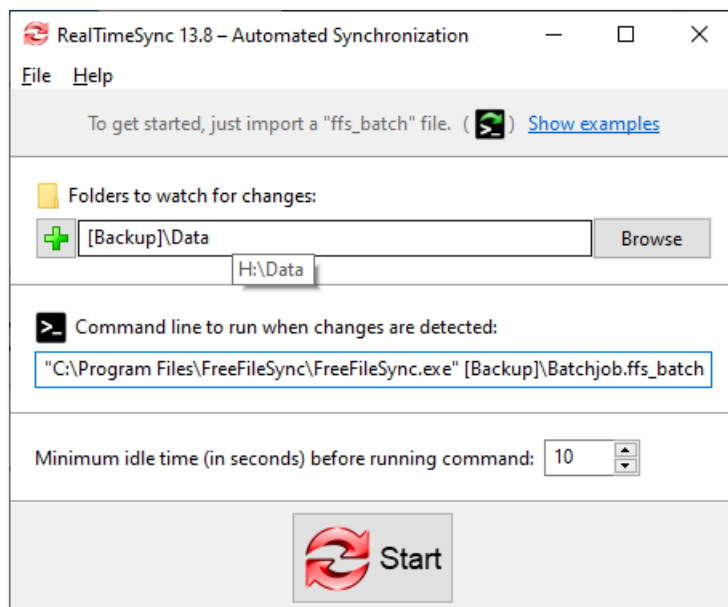
"Backup" is the volume name of the USB stick in our example.

Whenever directory H:\Data becomes available, RealTimeSync executes the command line which starts the batch job located on the stick. RealTimeSync will also trigger each time files are modified in H:\Data.

> **Note**
> The full path of the last changed file and the action that triggered the change notification (create, update or delete) are written to the environment variables `%change_path%` and `%change_action%`.
> They are **only visible** for the command line that RealTimeSync is executing.

Example: Log names of changed files and directories

Write a list of all changes to a log file: (Windows)

```
echo %change_action% %change_path% >> %csidl_Desktop%\log.txt
```

Write a list of all changes to a log file: (Linux/macOS)

```
echo $change_action $change_path >> ~/Desktop/log.txt
```

Limitations:

- If multiple changes happen at the same time, only the path of the first file is written to variable `%change_path%`.

- While RealTimeSync is executing the command line, monitoring for changed files is temporarily inactive.

- RealTimeSync relies on receiving change notifications from the operating system. In some cases it just doesn't receive any, e.g. a network path with badly-written/incomplete driver implementation. These buggy drivers often do not fail with an error code, but just do nothing.

The command line usually starts a synchronization task using FreeFileSync which naturally leads to additional file change notifications. Therefore, the RealTimeSync change detection has to be deactivated to not go into an endless loop. On the other hand, it is not likely that changes (other than those from FreeFileSync) happen in first place since RealTimeSync runs the command line only after the user-specified idle time has passed. In any case, files changed during the execution of FreeFileSync will be synchronized the next time FreeFileSync runs.

RealTimeSync: Run as Service (Windows)

RealTimeSync is designed to run as a background process which does not need further attention. Depending on your requirements, there are a number of ways to start it automatically. Generally, the goal is to execute a command line of the form:

```
<FreeFileSync installation folder>\RealTimeSync.exe <path to
*.ffs_real or *.ffs_batch file>
```

e.g.:

```
"C:\Program Files\FreeFileSync\RealTimeSync.exe" "D:\Backup
Projects.ffs_real"
```

Example: Run RealTimeSync on Windows login

Create a new shortcut, enter the command line from above as target and place it into the Windows autostart folder. (Enter **shell:startup** in the Windows Explorer address bar to find the folder quickly.)
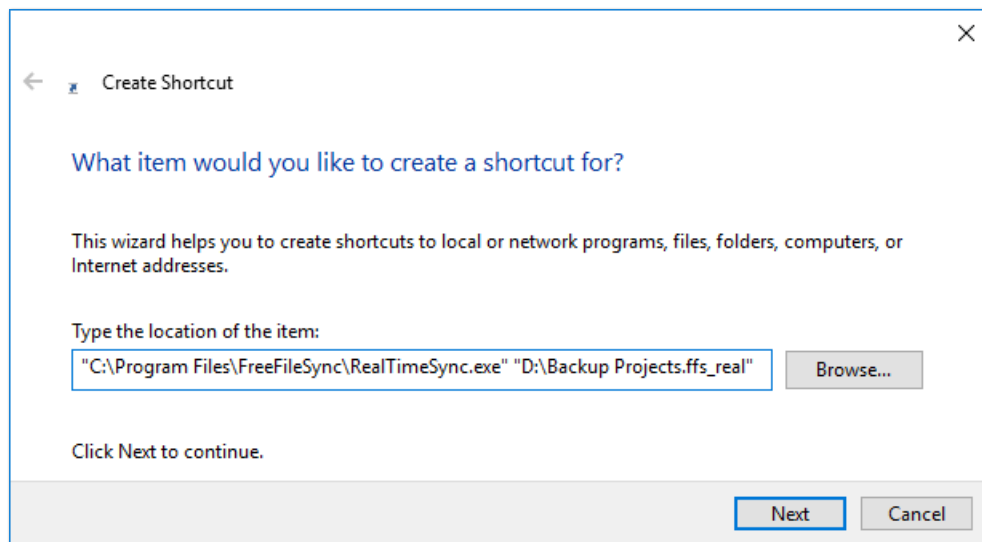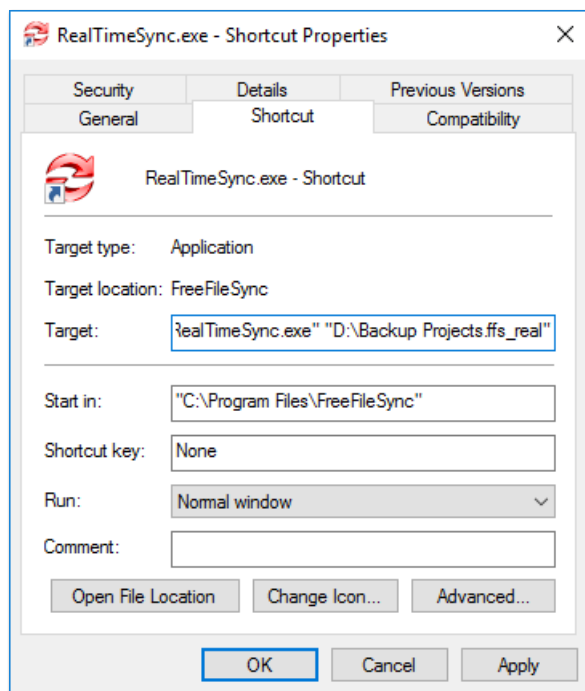
Example: Start RealTimeSync as a Service

RealTimeSync should be monitoring while Windows is running, irrespective of currently logged-in users: Create a new task in your operating systems's task scheduler and have it execute the command line above when the system starts. See **Schedule Batch Jobs** for an example of how to add a task. Then change the user which runs the task to **SYSTEM** - a special user account always running in the background.
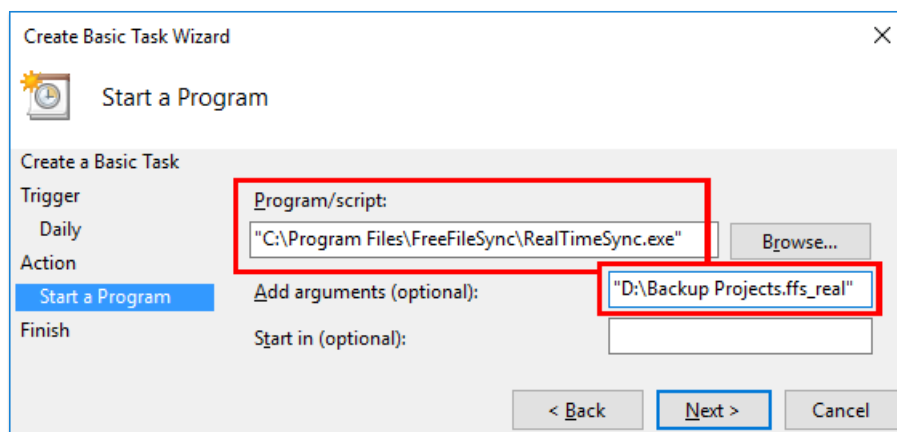
Schedule Batch Jobs

1. Create a new batch job via FreeFileSync's main dialog: **Menu → File → Save as a batch job...**
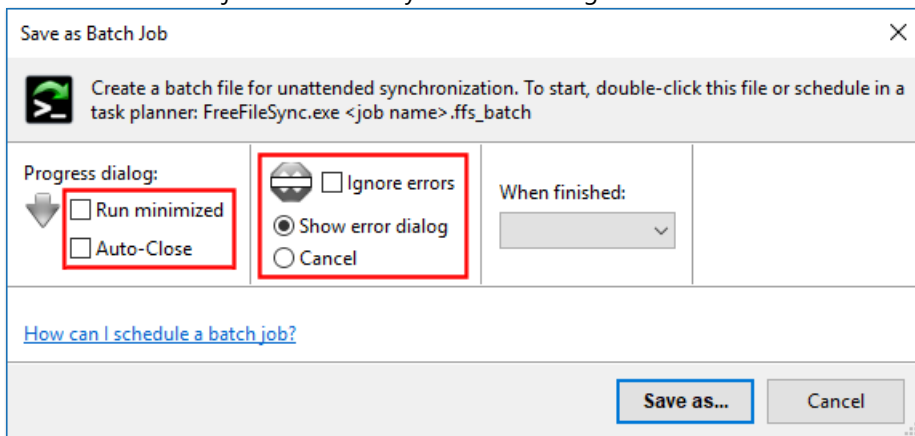
2. By default, FreeFileSync will show a progress dialog during synchronization andwill wait while the summary dialog is shown. If the progress dialog is not needed, enable checkbox **Run minimized** and also set **Auto-Close** if you want to skip the summary dialog at the end.

> **Note**
> Even if the progress dialog is not shown at the beginning, you can make it **visible at any time** during synchronization by double-clicking the FreeFileSync icon in the notification area.

3. If you don't want error or warning messages to stall synchronization when no user is available to respond, either check **Ignore errors** or set **Cancel** to stop the synchronization at the first error.

4. The FreeFileSync batch job can be started by double-clicking on the `ffs_batch` file or it can be set up in your operating system's scheduler:

   - **Windows**: Task Scheduler
   - **macOS**: Automator and Calendar
   - **Linux**/macOS: Cron Job

> **Note**
> Be sure to enable **Auto-Close** and **Ignore errors/Cancel** if you schedule the `ffs_batch` file to run under a **different user account**. With no one there to close the results dialog manually, the task would hang indefinitely.

Windows: Task Scheduler

5. Open the Task Scheduler either via the start menu, or enter `taskschd.msc` in the run dialog (keyboard shortcut: Windows + R).

6. Create a new **basic task** and follow the wizard.

7. Make **Program/script** point to the location of FreeFileSync.exe and insert the `ffs_batch` file into **Add arguments**.

8. Use quotation marks to protect spaces in path names, e.g. `"D:\Backup Projects.ffs_batch"`
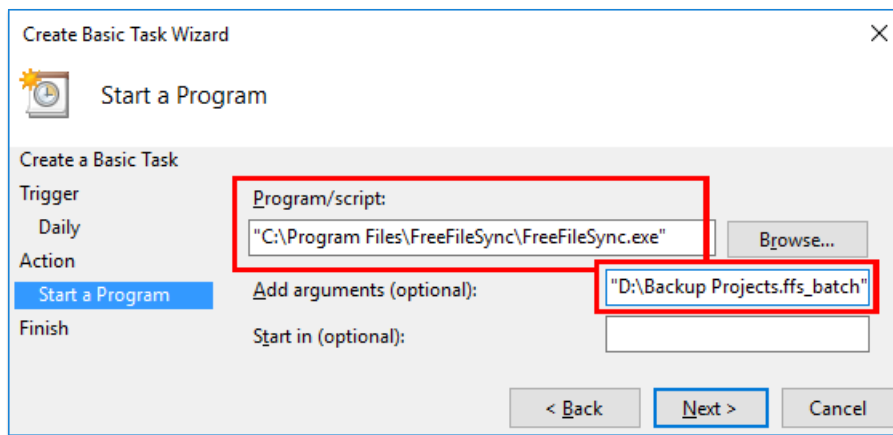
> **Note**
> - *Program/script* always needs to point to an executable file like FreeFileSync.exe even when the `ffs_batch` file association is registered. If an `ffs_batch` file was entered instead, the task would return with error code 2147942593 (0x800700C1), "%1 is not a valid Win32 application".
>
> - If you schedule FreeFileSync to run under a different user account, note that the configuration file `GlobalSettings.xml` will also be read from a different path, `C:\Users\<username>\AppData\Roaming\FreeFileSync`, or in the case of the SYSTEM account from
> `C:\Windows\System32\config\systemprofile\AppData\Roaming\FreeFileSync`.
> You can force usage of a particular `GlobalSettings.xml` file by passing it as a **Command Line** parameter.

macOS: Automator and Calendar

5. Open Launchpad and run **Automator**.
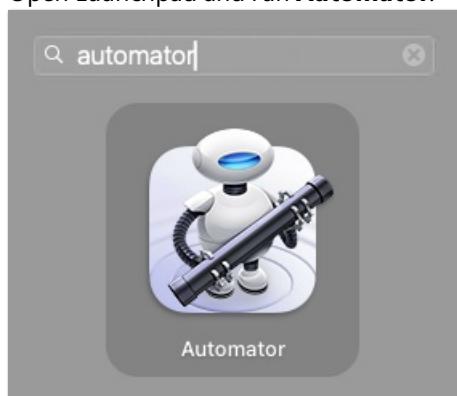
6. Create a new **Calendar Alarm**.
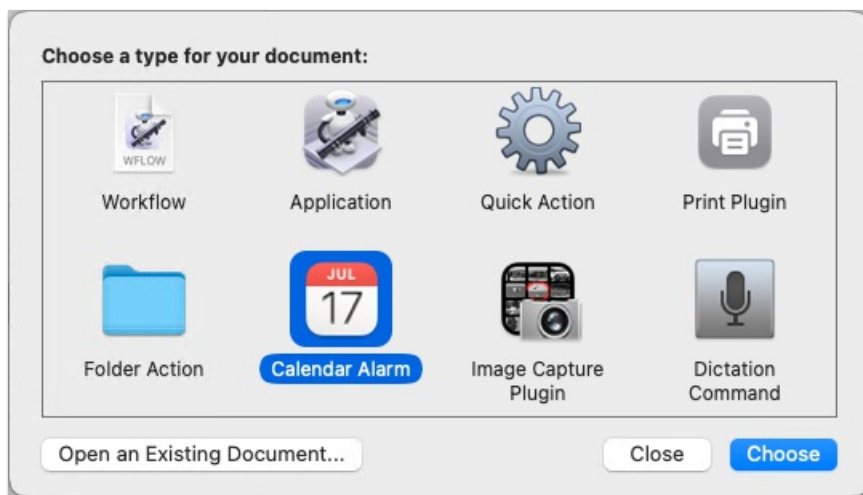
7. Drag and drop the `ffs_batch` file on the workflow panel.
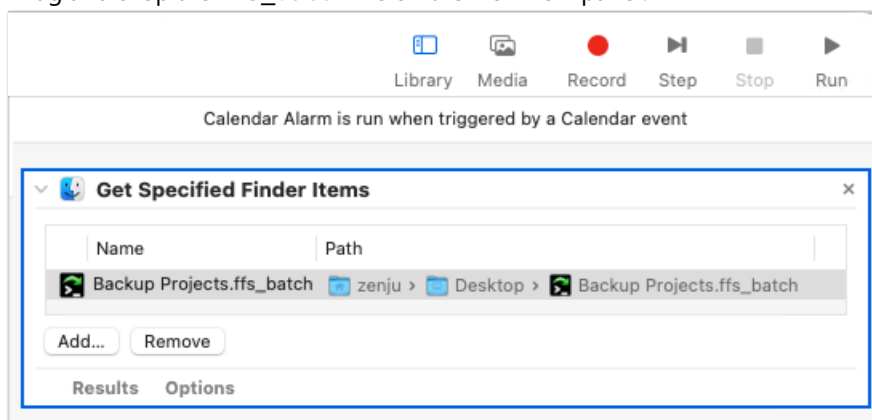
8. Double-click *Actions / Files & Folders / **Open Finder Items*** to add it to the workflow.
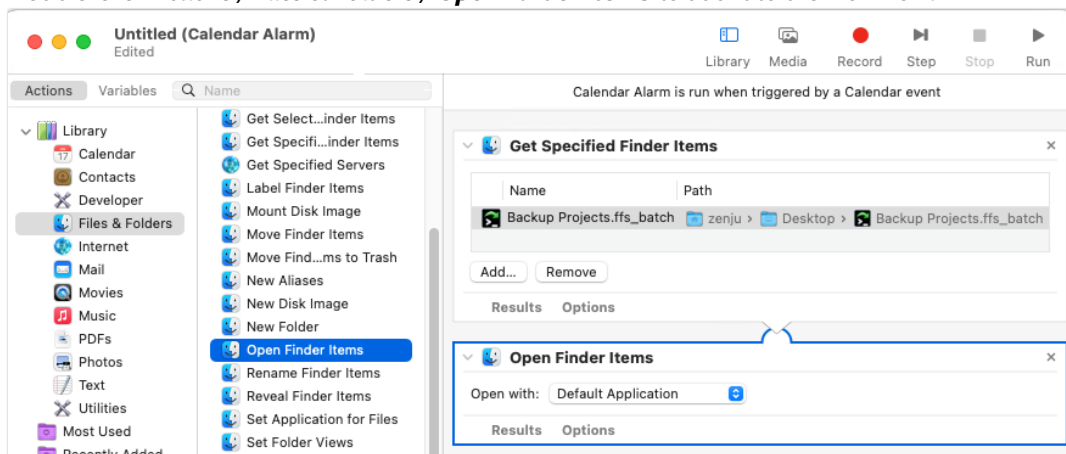
9. Go to **File → Save…** and save the Automator job.
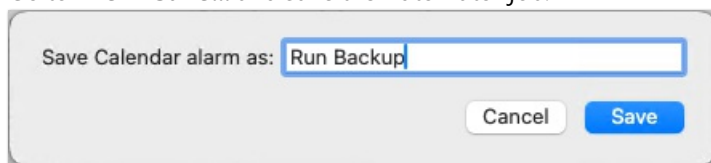
10. The Calendar app will start automatically with the Automator job scheduled to the current day. You can now select a different time for synchronization or make it a recurring task.
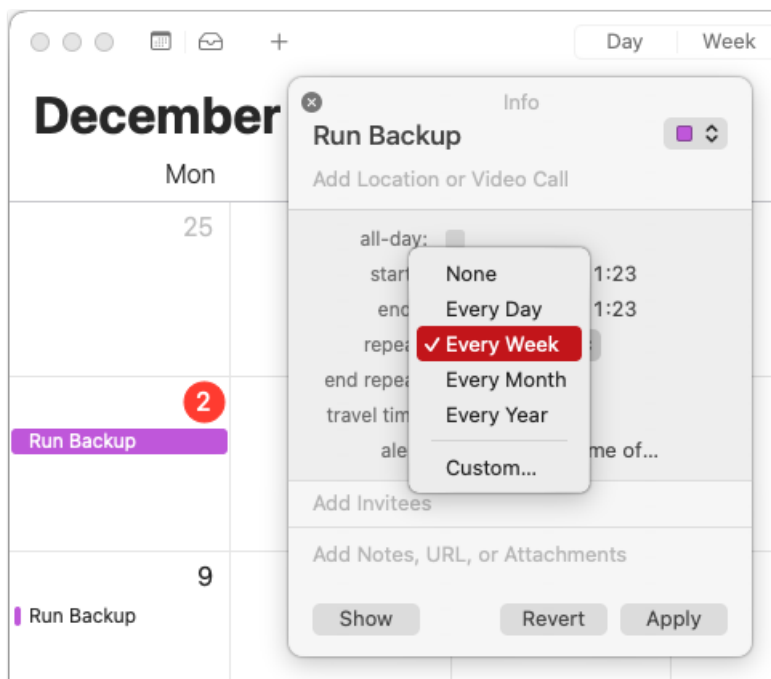
Linux/macOS: Cron Job

Cron executes arbitrary command lines repeatedly at given intervals.

To schedule a FreeFileSync batch job, construct a command line for cron consisting of the path to the FreeFileSync executable followed by the path to the FreeFileSync batch job, e.g.

```
/opt/FreeFileSync/FreeFileSync "/home/zenju/Backup
Projects.ffs_batch"
```

macOS: The executable is located inside the application package, e.g. for an all-users installation:

```
/Applications/FreeFileSync.app/Contents/MacOS/FreeFileSync
```

Open **cron**'s **tab**le of scheduled jobs for editing:

```
crontab -e
```

Each ctrontab line begins with conditions for recurring execution of the command line that follows. Cron's basic concept is to run a command *every minute* unless **limits** are applied:

To run only *every hour*, the *minute* must be fixed:

```
# minute   hour   day of month   month   day of week
    0        *          *           *          *
```

To run once *every day*, set both *minute and hour*; e.g. run daily at 17:00:

```
# minute   hour   day of month   month   day of week
    0        17         *           *          *
```

Multiple items are separated by "`,`", ranges specified using "`-`", and interval steps by "`/`".

```
# minute   hour   day of month   month   day of week
  */10     9-17         *           *        mon,fri
```

Example: Run every 10 minutes but only fom 9:00 to 17:00 and only on Monday and Friday

To run once after each *system startup*, use alternative syntax "**@reboot**":

```
@reboot    sleep 60; /opt/FreeFileSync/FreeFileSync
"/home/zenju/Backup Projects.ffs_batch"
```

Example: Start synchronization 60 seconds after system reboot

Synchronization Settings
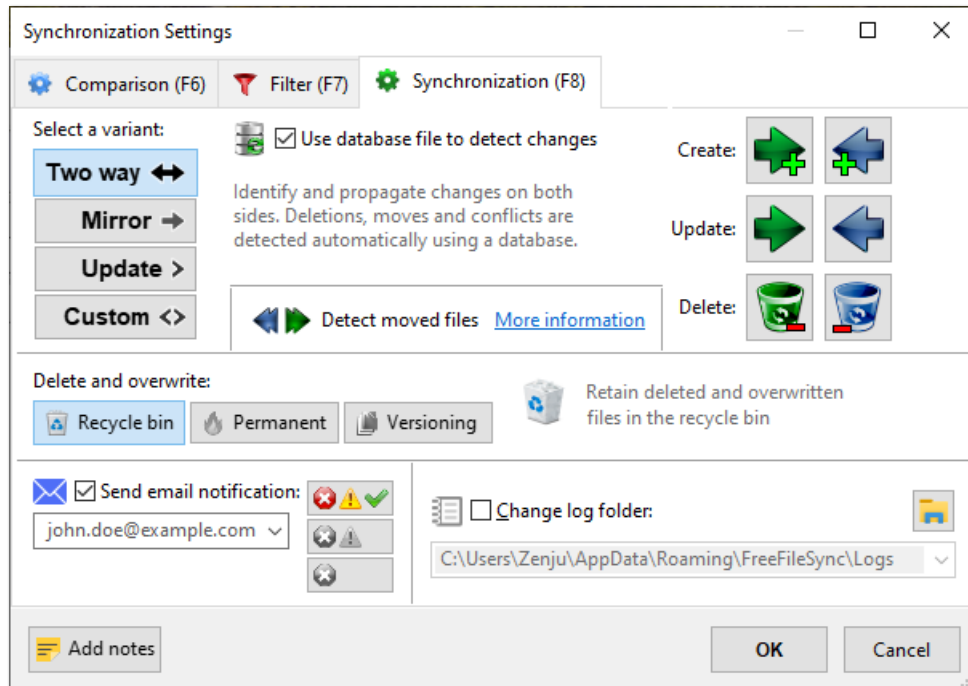
Synchronization Variants

There are three basic synchronization variants:

- If both left and right folders contain files you're working on, and you want changes (creates, updates, and deletes) to flow in both directions, then select **Two way**.
Database files ("`sync.ffs_db`") will be created after the first sync and be used to compare the current file system state against the last synchronization in order to determine the sync directions.

- If one folder contains your work files and the other is for backup, then select the **Mirror** variant. The left folder is the source and the right folder the target. The synchronization will create and delete files on the target as needed until it becomes an exact copy of the source.

- If you only want to add files to your backup, but never delete, then select the **Update** variant. Files deleted on the source side will *not cause file deletion* on the backup drive (e.g. after you've made room for new photos on a digital camera). On the other hand, files deleted on the backup drive will *not be copied over a second time* (e.g. after you have removed photos you don't want to keep).

In order to handle special synchronization scenarios you can also set up **Custom** rules. These can either be based on the categories determined after folder comparison (left/right only, left/right newer), or on detected changes (create, update, delete) if you select *Use database file to detect changes*.

Detect Moved Files
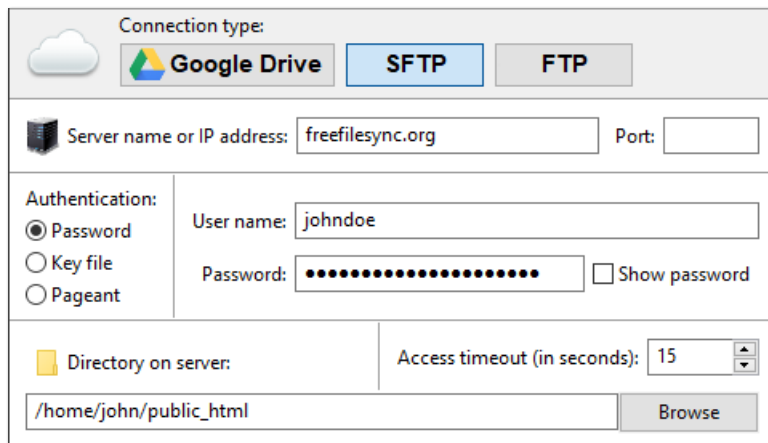
FreeFileSync is able to detect moved files and quickly apply the same move on the target side during synchronization instead of a slow copy and delete. To makes this work, **Use database file to detect changes** must be checked and the **file system must support file IDs**.

Note

- Detection of moved files is not yet possible when synchronizing a folder pair for the first time. Only beginning with the **second sync** the database files are available to determine moved files.

- Detection is **not supported** on file systems that **don't have (stable) file IDs**. Most notably, certain file moves on **FAT** file systems cannot be detected. Also, protocols like **SFTP** do not support move detection. In these cases FreeFileSync will automatically fall back to "copy and delete".

SFTP and FTP Setup

FreeFileSync supports synchronization with SFTP and FTP natively. Just enter your login information into the dialog shown for cloud folder selection:

> **Note**
>
> In case the (S)FTP server sets file **modification times** to the **current time** you can do a **Compare by File Size** as a workaround. Another solution is to set up the *Two way* variant and have the files with the newer dates be copied back from the server during the next synchronization.

Configure SFTP for Best Performance

By default, FreeFileSync creates one connection to the server and uses one SFTP channel, i.e. only a single SFTP command can be sent and received at a time. Since most of this time is spent waiting due to the high latency of the remote connection, you can speed up reading large folder hierarchies by increasing both the connection and channel count.

*The folder reading time is reduced by a factor of **N** x **M** when using N connections with M channels each.*

Example: 10 connections using 2 channels each can yield a **20** times faster folder reading.

- The creation of additional connections and channels takes time. If you are only scanning a small remote folder, setting up too many connections and channels might actually slow the overall process down. Creating extra connections is slower than creating extra channels.

- SFTP servers have internal limits on the number of allowed connections and channels. Generally, servers expect one connection per user, so this number should be kept rather low. If too many connections and channels are used, the server may decide to stop responding.

- Unlike connections, additional SFTP channels are (currently) only used during folder reading (comparison), but not during synchronization.

- **Enable compression** to improve performance if the connection to the SFTP server is *slow* and the data is mostly *uncompressed* (e.g. copying text files over a slow internet connection). However, if the connection is very fast (e.g. a local network), or the data is already compressed (e.g. zip files), the CPU overhead of the zlib compression algorithm *might slow transfer times down* and the option is better left unchecked.

> **Advice**
> Start with low numbers and make tests with different combinations of connections and channels for your particular SFTP synchronization scenario to see what gives the highest speed. Note, however, that FreeFileSync **reuses existing** SFTP connections/channels. Therefore, you should **restart** FreeFileSync before measuring SFTP speed.

Tips and Tricks

Change settings with a single mouse click: Press and hold the right mouse button until the context menu is shown, then release while over the selection:
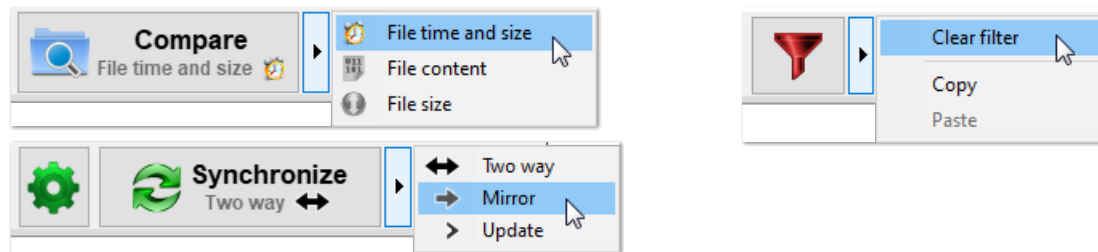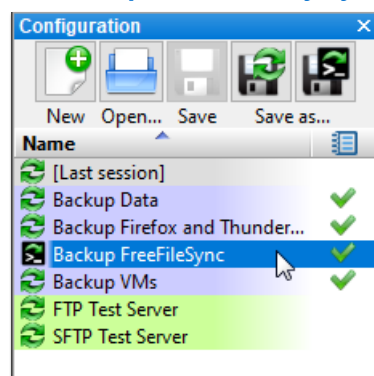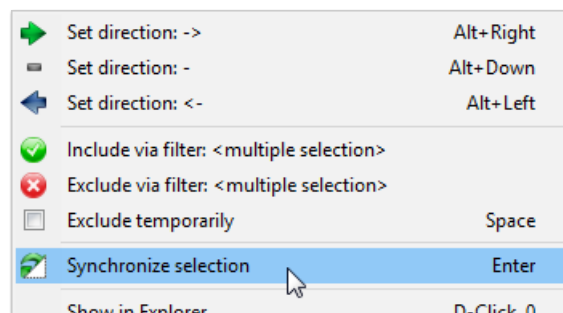

Select multiple configurations at a time:
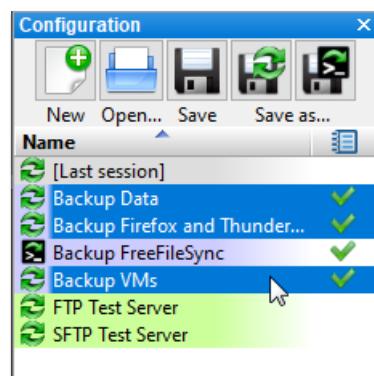
Select multiple items via mouse, and refine the selection by holding the Control key while clicking.
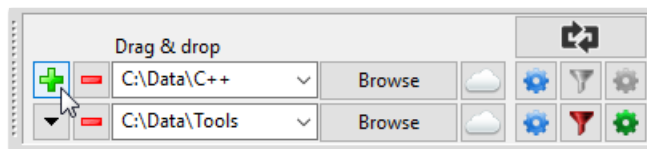
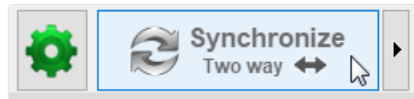## Start comparison directly by double-clicking on a configuration:

Run a partial synchronization only for the currently selected files:

## Synchronize multiple folder pairs at a time with different configurations:

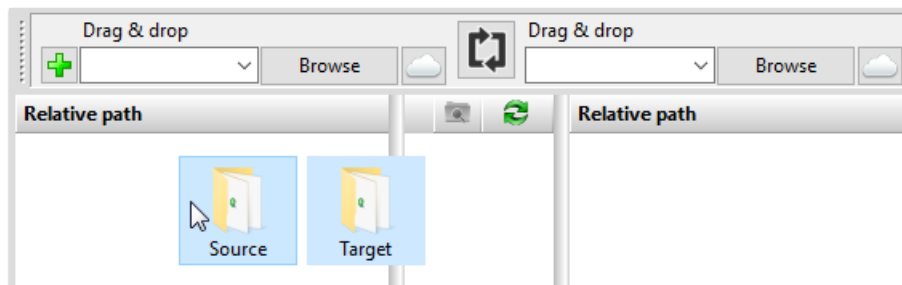**Start synchronization directly without clicking on compare first:**

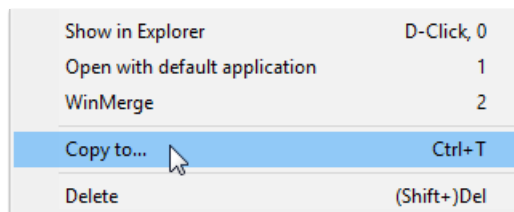**Move a window by clicking on a free area and holding the mouse button:**

Open a batch configuration for edit via the Windows Explorer context menu:

☑ BatchRun.ffs_batch
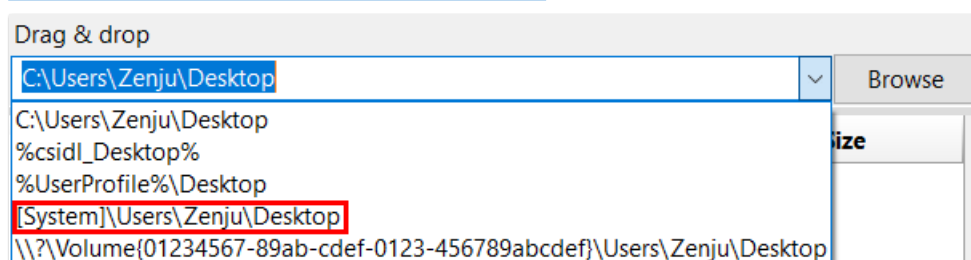| **Open** |
| Edit with FreeFileSync |
| Open with... |
| Share with ▶ |
| Send to ▶ |
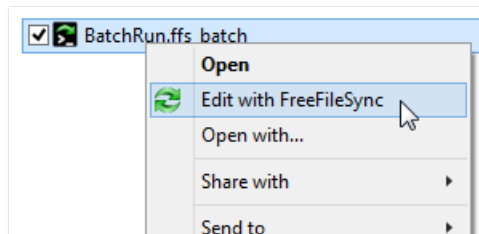
Drag and drop two folders at a time from Windows Explorer to fill a folder pair in one go:

**Copy files selected on the main dialog to an alternate folder and thereby save a "diff":**

Show in Explorer	D-Click, 0
Open with default application	1
WinMerge	2
Copy to...	Ctrl+T
Delete	(Shift+)Del

Use a volume name instead of a drive letter:

Drag & drop

C:\Users\Zenju\Desktop

C:\Users\Zenju\Desktop
%csidl_Desktop%
%UserProfile%\Desktop
[System]\Users\Zenju\Desktop
\\?\Volume{01234567-89ab-cdef-0123-456789abcdef}\Users\Zenju\Desktop
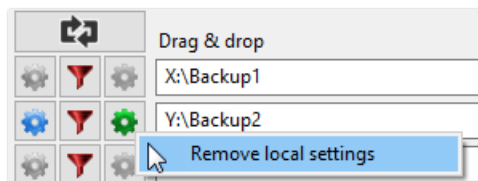
Show thumbnail icons via the column header context menu:
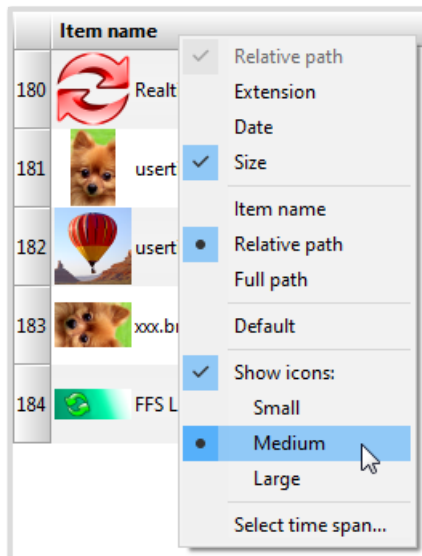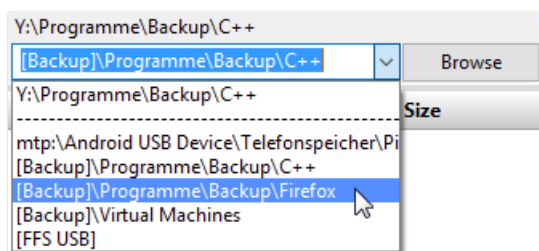

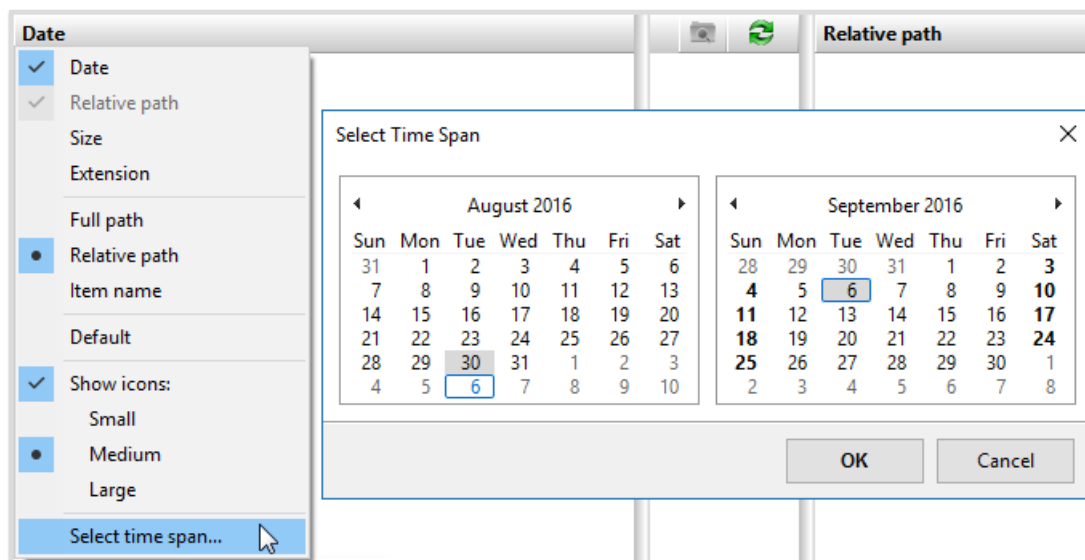## Save the current view filter selection as default:
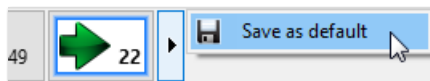
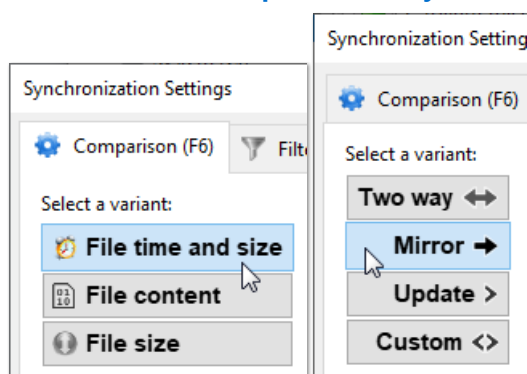## Remove local settings from individual folder pairs:

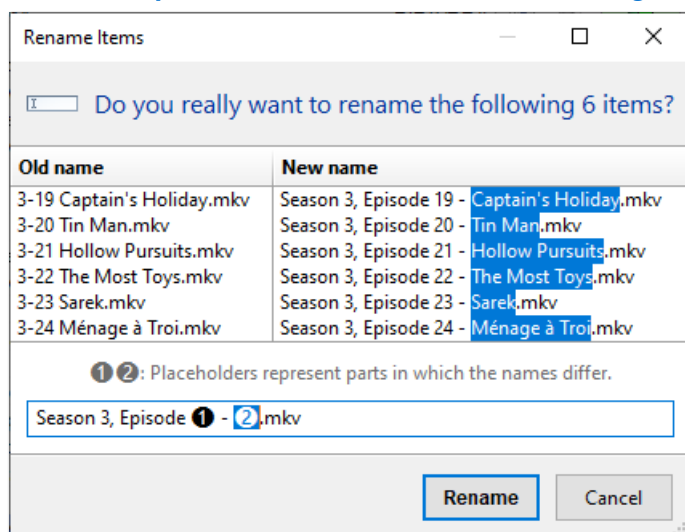## Remove obsolete paths from the folder drop-down by using mouse hover and Delete key:

Select a time span for files to include via the date column context menu:

Double-click on comparison and synchronization variants to confirm the dialog:

Select multiple files and rename all of them in one go (via context menu or F2 key):

Variable Drive Letters

USB memory sticks or external hard disks often get different drive letters assigned than the last time when they were plugged into the computer. FreeFileSync offers the following solutions:

Option 1: Specify a path by using a **unique volume name** instead of a drive letter:

```
E.g. [Backup-Disk]\folder
```

replaces `E:\folder` when the name of the USB stick in drive `E:\` is "Backup-Disk".
You can change a volume name by right-clicking on the drive letter in Windows Explorer and selecting **Rename**.

Option 2: Refer to a disk volume by its **unique GUID**:

```
E.g. \\?\Volume{01234567-89ab-cdef-0123-456789abcdef}\folder
```

A volume GUID uniquely identifies a particular drive partition and will not change over time. To get an overview of all volume GUIDs for mounted drives on the system you can run the**mountvol** command line tool.

Note
It is **not required to look up and enter** a volume name, volume guid or macro manually.
Simply select an alternative path from the drop down menu:
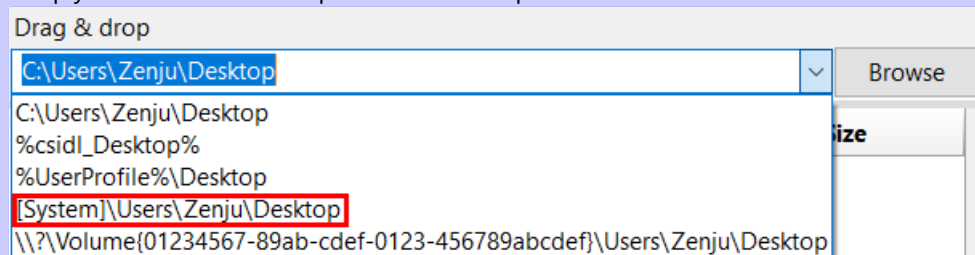
File Versioning

When you need to preserve files that have been deleted or overwritten, it's often sufficient to select **Recycle bin** in synchronization settings. However, this is only available for local drives and offers little control on how to store and how long to keep the files. FreeFileSync therefore has an additional option, **Versioning**.

1. Keep only the most recent versions

In synchronization settings, set deletion handling to **Versioning** and naming convention to **Replace**. Deleted files will be moved to the specified folder without any decoration and will replace already existing older versions.
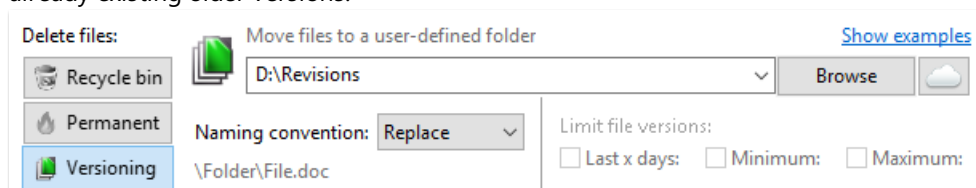
2. Keep multiple versions of old files

A. Set deletion handling to **Versioning** and naming convention to **Time stamp [File]**. FreeFileSync will move deleted files into the provided folder and add a time stamp to each file name. The structure of the synchronized folders is preserved so that old versions of a file can be conveniently accessed via a file browser.

   Example: Last versions of the file `Folder\File.txt` inside folder `D:\Revisions`

   ```
   D:\Revisions\Folder\File.txt 2020-12-11 111111.txt
   D:\Revisions\Folder\File.txt 2020-12-12 122222.txt
   D:\Revisions\Folder\File.txt 2020-12-13 133333.txt
   ```

B. With naming convention **Time stamp [Folder]** files are moved into a time-stamped subfolder of the versioning folder while their names remain unchanged. This makes it easy to manually undo a synchronization by moving the deleted files from the versioning folder back to their original folders.

   Example: Last versions of the file `Folder\File.txt` inside folder `D:\Revisions`

   ```
   D:\Revisions\2020-12-11 111111\Folder\File.txt
   D:\Revisions\2020-12-12 122222\Folder\File.txt
   D:\Revisions\2020-12-13 133333\Folder\File.txt
   ```

3. Save versions at certain intervals

With naming convention **Replace** it is possible to refine the granularity of versions to keep by adding **Macros** to the versioning folder path. For example, you can save deleted files on a daily basis by adding the `%date%` macro:

Example: Last versions of the file `Folder\File.txt` inside folder `D:\Revisions\%date%`

```
D:\Revisions\2020-12-11\Folder\File.txt
D:\Revisions\2020-12-12\Folder\File.txt
D:\Revisions\2020-12-13\Folder\File.txt
```

Volume Shadow Copy (Windows only)

FreeFileSync supports copying locked or shared files by creating a Volume Shadow Copy of the source drive. This feature can be configured via **Menu → Tools → Options: Copy locked files**.

> **Note**
> - The volume snapshot created by the Volume Shadow Copy Service is only used for copying files that are actually locked.
>
> - Accessing the Volume Shadow Copy Service requires FreeFileSync to be started with administrator rights.