

## Folder apps

### Description

This is a bundle of six apps that do interesting things with folders.

- [Filetypes Counter](#) counts the files in a job folder per filetype and adds these counters and some extra info to the job folder as private data and dataset.
- [Delete Files](#) deletes files from a job folder.
- [Hot Folder Monitor](#) checks a certain folder every so often to see if files disappear from the folder. If not, it means that the application that should be processing those files is not running.
- [File Pool Cleanup](#) allows to remove files from a file pool that are older than a certain number of days.
- [File Sorter and Spacer](#) puts files inside a job folder in a certain alphabetical order based on a property of the file, and optionally injects them as new jobs spaced apart.
- [Submit Filespec](#) can be used to submit files from a hierarchy, but only those that match a certain file specification.

### Compatibility

Switch 2021 Fall and higher. Windows or Mac OSX.

### Compatibility third-party applications

There are no dependencies on third-party applications.

## Filetypes Counter



*Filetypes Counter* counts the files in a job folder per filetype, and adds these counters and some extra info to the job folder as private data and dataset.

For each filetype private data will be added with the number of files.

The name of the private data is built like this: <key>.<filetype>.count

Private data will always be added for all the filetypes you define in the properties, even if there is no file of that filetype, so that you can use this information further in your flow.

The other files will be counted together in <key>.others.count, unless you choose to add separate private data for each filetype found. An extra 'noExtension' filetype will be added if necessary.

Depending on your OS, some invisible files can be present in the job folder, so these special filetypes are always counted as well: startingWithDot, .DS\_Store, and thumbsdb.

Extra stats private data will always be added:

- <key>.folders.count: number of subfolders found
- <key>.stats.count: count of all files
- <key>.stats.hidden: count of startingWithDot, .DS\_Store, and thumbsdb files
- <key>.stats.visible: count - hidden
- <key>.stats.folderLevels: number of sublevels found in the job folder

A dataset will also be attached to the job with files names and paths on top of the info above. See screenshot on next page.

## Connections

*Filetypes Counter* has incoming connections and a single outgoing connection. Jobs that are not folders pass unchanged.

## Properties detailed info

Property	Value
<b>Element type</b>	
<b>Name</b>	Filetypes Counter
<b>Description</b>	
<b>Subfolder levels</b>	All
<b>Filetypes</b>	File types defined
<b>Count other found filetypes</b>	Together (others)
<b>Private data key prefix</b>	filetypes
<b>Dataset name</b>	filetypes

### Flow element properties

- **Subfolder levels:** Which subfolder level(s) should be scanned?  
 All, Root level only or Custom  
 If Custom: From (top level), *considering root level = 1*  
 To (bottom level), *must be greater than or equal to From (top) level*
- **Filetypes:** Select the filetypes to count.  
 If custom, one per line, format must be of type \*.pdf
- **Count other found filetypes:**
  - Together (others)
  - Add separate private data
- **Private data key prefix:** name of private data key prefix, default 'filetypes'
- **Dataset name:** name of dataset, default 'filetypes'

### Sample result

Metadata dataset

Embedded

External

filetypes

Location path syntax

XMP location path

XML location path

JDF location path

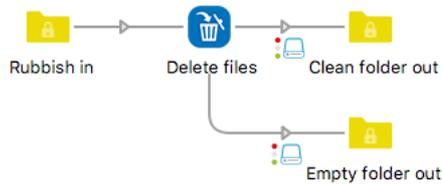
XPath expression

XML location path

XML data tree

Name	Value
▼ filetypes	
▶ startingWithDot	
▶ DS_Store	
▶ thumbsdb	
▶ folders	
▼ others	
count	5
▶ item	
▼ item	
id	5
level	2
name	ProximaNova-Light.otf
relativePath	Document fonts/ProximaNova-Light.otf
▶ noExtension	
▼ invisible	
count	1
▶ item	
▶ indd	
▶ idml	
▶ xml	
▼ stats	
count	12
hidden	1
visible	11
folderLevels	2

## Delete Files



This app deletes certain files from a job folder. This can be done by specifying the files that have to be kept, for example, keep all PDF files and delete the rest, or inversely by specifying the files that have to be deleted, for example, delete all files that do not start with a number.

The specification of files that have to be kept or deleted can be done in three ways: by file types, by file patterns, or by regular expressions. The three methods can be combined.

There is an additional property to remove hidden files.

When the job is not a folder it is ignored and passed along the outgoing connection.

### Connections

*Delete Files* has outgoing traffic-light connections: Success and Error. The error connection is used in case the criteria you have defined for deleting the files would delete ALL files from the job folder. This of course does not make sense. In all other cases, the job folder is sent along the success connection, also when no files have been deleted.

## Properties detailed info

Property	Value
Element type	Delete files
<b>Name</b>	Delete files
<b>Description</b>	
<b>Delete or keep files</b>	Keep
<b>Remove hidden files</b>	Yes
<b>File types</b>	File types defined
<b>File patterns</b>	None
<b>Regular expressions</b>	None

### Flow element properties

- **Delete or keep files:** this is a drop-down of “Delete” and “Keep”. It determines whether the files matched by the properties below will be deleted or kept. The default is “Delete”.
- **Remove hidden files:** files that start with a dot are considered hidden files and are not always shown in the Finder/Explorer. When choosing “Yes” they are removed.
- **File types:** here you define the file types that have to be deleted or kept using the file types dialog. Default: None.
- **File patterns:** here you can define file patterns using wildcards, ? standing for 1 unknown character, and \* for any number of unknown characters. There can be multiple file patterns, one per line. Default: None.
- **Regular expressions:** this property works in the same way as the file patterns property, except of course that here you define multiple regular expressions, one per line. Default: None.

The three methods can be combined. When the “Delete” option is chosen a file is marked for deletion as soon as it matches one of the criteria. When the “Keep” option is chosen a file is marked for deletion if it matches none of the criteria.

## Hot Folder Monitor



*Hot Folder Monitor* checks a certain folder structure every so often and remembers the list of files that are present in that folder structure. When there are files that have not disappeared the next time around an HTML file is created listing those files. This file can then be used to trigger a mail to alert somebody to the fact that the application that should be processing the files from the folder may not be running. The app also sets the e-mail body to the same contents. This is convenient if you prefer to put the list of files in the mail body rather than attach the text file.

For the HTML-savvy people: there is a title enclosed in `<p class='hot-folder-monitor'>` and a list of file paths as an unordered list `<ul class='hot-folder-monitor'>`, so you can add this class to the CSS of the HTML template used in “Mail send” to format the text to your liking.

Some hot folder-based systems have a configuration file in the hot folder and that file will of course always be there. There are three properties with which files to be ignored can be specified.

When the app sees that all files have disappeared, no text file is created.

The app checks the hot folder recursively. In other words, when all the hot folders monitored by an application share the same root folder, it is not necessary to create an app instance for every single folder, it is enough to monitor the root folder.

### Connections

*Hot Folder Monitor* does not have any input connections. It has a single output connection along which the text file is sent when necessary.

## Properties detailed info

Property	Value
Element type	Hot Folder Monitor
<b>Name</b>	Hot Folder Monitor
<b>Description</b>	
<b>Hot folder to be monitored</b>	/Users/fp/Desktop/tmp <input type="text"/>
<b>Interval (secs)</b>	300
<b>Exclude file types</b>	None
<b>Exclude file patterns</b>	None
<b>Exclude regular expressions</b>	None

### Flow element properties

- **Hot folder to be monitored:** choose the folder to be monitored. There is no default.
- **Interval (secs):** make sure to choose a value that is higher than that used by the application that scans the monitored folder. Also allow for some extra processing time to reduce the chance of false alarms. Default: 300. Note that when the application is not running for a longer period the app will trigger the notification every time it runs!
- **Exclude file types:** Default: None.
- **Exclude file patterns:** Default: None.
- **Exclude regular expressions:** Default: None.

## File Pool Cleanup



*File Pool Cleanup* checks a certain root folder every so often and scans it and all its subfolders looking for files that are older than a certain period. These files are removed from their location and injected into the flow for further processing, usually deleting them or archiving them in a different location. One use case for this tool is to clean up the location where you store customer files in order to be GDPR-compliant.

To avoid files being removed from important locations, the definition of the root folder as / on Mac, or C: on Windows is not allowed! If you want to use other root folders or network drives, you must allow this specifically.

Another safety measure is that you can specify the maximum number of files to be processed. When the limit is passed the files are not removed but instead a log file is created containing a list of the files that should have been removed. During the setup phase it is therefore a good idea to set the limit to 1. This will most probably generate a log file that you can analyze to verify you are using the correct settings. When everything is OK, you can set the limit to 0 which stands for no limit.

There is a choice of selecting files based on the modification date or the creation date, and two properties to define certain files and/or folders that must not be removed.

Finally, there is an option to remove empty folders. Note that only folders that are empty when the app runs are removed. Folders that become empty as a result of the cleanup are not immediately removed, but they will be the next time.

The app injects the files as new jobs into the flow and then deletes the original file. This can of course fail when Switch does not have sufficient access rights. In that case there is also an error logfile that is generated. It is a very simple HTML. The same content is also added as the job's mail body so it can easily be used in a "Mail send".

For the HTML-savvy people: there is a title enclosed in `<p class='file-pool-cleanup'>` and a list of file paths as an unordered list `<ul class='file-pool-cleanup'>`, so you can add this class to the CSS of the HTML template used in "Mail send" to format the text to your liking.

The app also sets the hierarchy path of the job.

## Connections

*File Pool Cleanup* can be used with an incoming connection or without it. Without it, the cleanup is triggered at regular time intervals; with it, you can trigger the cleanup at specific moments. The app has outgoing traffic-light connections, a Data Success one for the files that have been cleaned up from the file pool, and a Log Error one to which the log file is sent in case the limit has been exceeded.

## Properties detailed info

Property	Value
Element type	File Pool Cleanup
<b>Name</b>	File Pool Cleanup
<b>Description</b>	
<b>Root folder</b>	/Users/fp/Desktop/tmp <span>...</span>
<i>Allow root and network folders</i>	No
<b>Scan interval</b>	1
<i>Unit of time</i>	Days
<b>Remove files older than</b>	31
<i>File date type</i>	Modification date
<i>Unit of time</i>	Days
<b>Remove empty folders</b>	No
<b>Ignore files</b>	None
<b>Ignore folders</b>	None
<b>Maximum number of files</b>	0
<b>Attach hierarchy info</b>	No

## Flow element properties

- **File pool root folder:** choose the root folder to be cleaned up. There is no default.
  - **Allow root and network folders:** when set to No, there is a check that the chosen root folder is not a drive letter or some folder on a network drive. This is to avoid whole disks being accidentally erased. Note that / on Mac and C: on Windows are never allowed for obvious reasons. Any other choice is at your own responsibility!
- **Interval:** Default: 1.
  - **Unit of time:** there is a choice of Days (default), Hours and Minutes.
- **Remove files older than:** Default: 30.
  - **File date type:** this is a drop-down of “Modification date” and “Creation date”. Default: Modification date.
  - **Unit of time:** there is a choice of Days (default), Hours and Minutes.
- **Remove empty folders:** Yes or No, default: Yes.
- **Ignore files:** This property allows the use of the File types and File patterns editors for defining file specifications that should never be cleaned. Default: None.
- **Ignore folders:** This property allows the use of the Folder patterns editor. Default: None.

- **Maximum number of files:** When the number of files eligible for cleanup is higher than this number, the files are not removed, but logged in a file that is sent along the log error connection. Default: 0.
- **Attach hierarchy info:** when set to “Yes” the hierarchy path to the file that was cleaned up will be added to the job ticket.

## File Sorter and Spacer



There is an element *Sort files in job* that sorts files in a job based on the file name. The sorting is achieved by adding a prefix to the file names inside the job folder. If you have a list of files that are alphabetically ordered like this:

abc\_2.pdf

def\_4.pdf

ghi\_1.pdf

jkl\_3.pdf

and you use the number at the end for sorting the files, the result will look like this:

1\_ghi\_1.pdf

2\_abc\_2.pdf

3\_jkl\_3.pdf

4\_def\_4.pdf

By adding the prefix, the files are also alphabetically sorted in the desired order.

This app works similarly but it cannot only use the file name, but also a file property. The properties currently supported are the file creation date, the file modification date, the file size, the number of pages and the page surface of the first page.

Another difference is that the app allows to sort in ascending or descending order.

And a final difference is that the app allows to inject the files and first-level folders inside the job folder as individual jobs with a time space in between. This can be interesting when the contents of the job folder must be handed off to a hot folder-based system and you want to ensure that this is done in specific order.

The app only scans the root folder of the job. Subfolders, if any, are treated as files. The sorting on the number of pages and the page surface only works for PDF and images files. Other file types will be treated as having 0 pages and 0 surface, meaning they will come out on top.

### Connections

*File Sorter and Spacer* has incoming connections and a single outgoing connection. Jobs that are not folders pass unchanged.

## Properties detailed info

Property	Value
<b>Element type</b>	File Sorter and Spacer
<b>Name</b>	File Sorter and Spacer
<b>Description</b>	
<b>Sort property</b>	File modification date
<b>Sort order</b>	Ascending
<b>Space files in job folder</b>	Job folder after job folder
<b>Space interval (secs)</b>	5
<b>Remove sequence prefix</b>	Yes

### Flow element properties

- **Sort property:** a drop-down list of
  - File creation date
  - File modification date
  - File size
  - File name
  - Number of pages
  - Page surface
 Default: File modification date
  
- **Sort order:** a drop-down of “Ascending”, “Descending”. Default: Ascending.
  
- **Space files in job folder:** a drop-down of “None”, “Job folder after job folder”, “Multiple job folders at the same time”. Default: None. When the value is to process job folder after job folder, all files of all waiting job folders trickle through one file at a time. When multiple job folders are processed simultaneously there will be one file from every waiting job folder that is injected as a new job at every space interval.
  - **Space interval (secs):** the default value is 5.
  - **Remove sequence prefix:** the files in the job folder are sorted by adding a prefix. Setting this option to “Yes” ensures that this prefix is removed when a file is injected into the flow as a new job.

## Submit Filespec



*Submit Filespec* works similarly to *Submit hierarchy*, but of course somewhat differently. *Submit hierarchy* allows you to ignore certain folders, but it always processes all the files in the folders that should be processed. This app allows to define file name characteristics for submitting files, e.g. only PDF files, or only files that start with a number, etc.

The file name specifications can be defined as file types, file patterns, and regular expressions. All three can be used at the same time: when a file matches one of the criteria, it is submitted into the flow.

Similar to *Submit hierarchy* this app also sets the hierarchy of the input file.

Finally, there is an option to remove empty folders. Note that only folders that are empty at the moment the app scans the hierarchy are removed. Folders that become empty as a result of all files in the folder having been submitted are not immediately removed, but they will be the next time the app scans the hierarchy and they are still empty.

### Connections

*Submit Filespec* does not have any incoming connections, and it has a single outgoing connection.

## Properties detailed info

Property	Value
<b>Element type</b>	Submit Filespec
<b>Name</b>	Submit Filespec
<b>Description</b>	
<b>Path</b>	/Users/fp/Desktop/tmp
<b><i>Allow root folders and network drives</i></b>	No
<b><i>Local copy mode</i></b>	No
<b>Scan interval (secs)</b>	30
<b>File types</b>	File types defined
<b>File patterns</b>	None
<b>Regular expressions</b>	None
<b>Remove empty folders</b>	No
<b>Attach hierarchy info</b>	Yes

### Flow element properties

- **Path:** choose the root folder of the hierarchy to be scanned. There is no default.
  - **Allow root folders and network drives:** when set to No, there is a check that the chosen root folder is not a drive letter or some folder on a network drive. This is to avoid whole disks being accidentally erased. Note that / on Mac and C: on Windows are never allowed for obvious reasons. Any other choice is at your own responsibility!
- **Scan interval (secs):** Default: 30.
- **File types:** here you define the file types that have to be submitted into the flow using the file types dialog. Default: None.
- **File patterns:** here you can define file patterns using wildcards, ? standing for 1 unknown character, and \* for any number of unknown characters. There can be multiple file patterns, one per line. Default: None.
- **Regular expressions:** this property works in the same way as the file patterns property, except of course that here you define multiple regular expressions, one per line. Default: None.
- **Remove empty folders:** Yes or No, default: Yes.
- **Attach hierarchy info:** when set to “Yes” the hierarchy path to the file that was injected will be added to the job ticket.

## What's new in v3

For this version all apps have been ported to NodeJS. At the same time all known feature requests have been implemented as well as some additional functionality.

- Filetypes Counter: the information gathered about the filetypes is now also added as an XML dataset
- File Pool Cleanup: allow network folders, ignore certain files and folders, allow triggering by an incoming job
- Submit Filespec: allow network folders, allow a local copy to check that a file has been correctly removed from its source location before injecting it
- File Spacer and Sorter: allow file spacing when injecting the files as new jobs
- Hot Folder Monitor: allow the exclusion of certain files and folders, the hot folder root is now scanned recursively
- Delete files: there is now a separate property to delete hidden files