Folder apps

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.

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.

Filetypes Counter Connections:

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

Filetupes Counter Properties:

Property	Description

Subfolder levels	Determines which subfolder level(s) should be scanned:
	
	• All
	• Root level only
	• Custom: this option triggers 2 subproperties:
	
	    • **From (top level)** considering root level = 1
	
	    • **To (bottom level)** must be greater than or equal to From (top level)

Filetypes	Filetypes that must be counted. If custom, one per line, format must be of type *.pdf
Count other found filetypes	Options are: • Together (others) • Add separate private data
Size unit	Unit in which the file size is expressed. Options are: • Automatic (the most suitable unit will be used, e.g. 500 bytes and not 0.0005 MB) • KB • MB • GB • TB
Size decimals	Determines the number of digits after the decimal point. Example: • Size decimals = "2": 34.06 KB • Size decimals = "5": 34.06000 KB
Private data key prefix	Name of the private data key prefix. Default is 'filetypes'. Example: If this property is set to 'filetypes', the private data will be stored in filetypes.folders.count, filetypes.stats.count, etc.

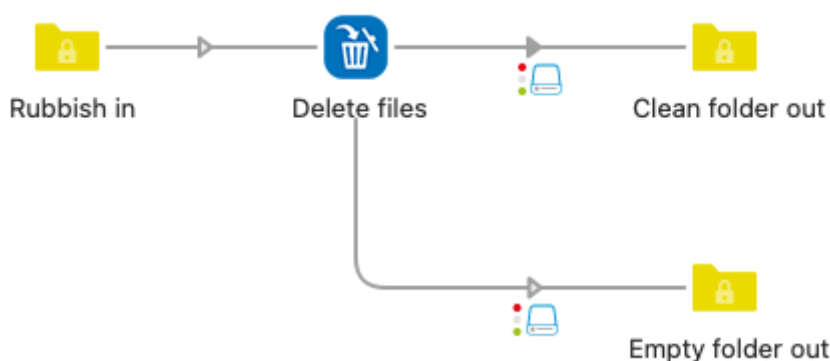| Property	Description
Dataset name	Name of the dataset. Default is 'filetypes'
Dataset model	JSON (default) or XML

Example of the external dataset in XML format (dataset name is 'filetypes'):
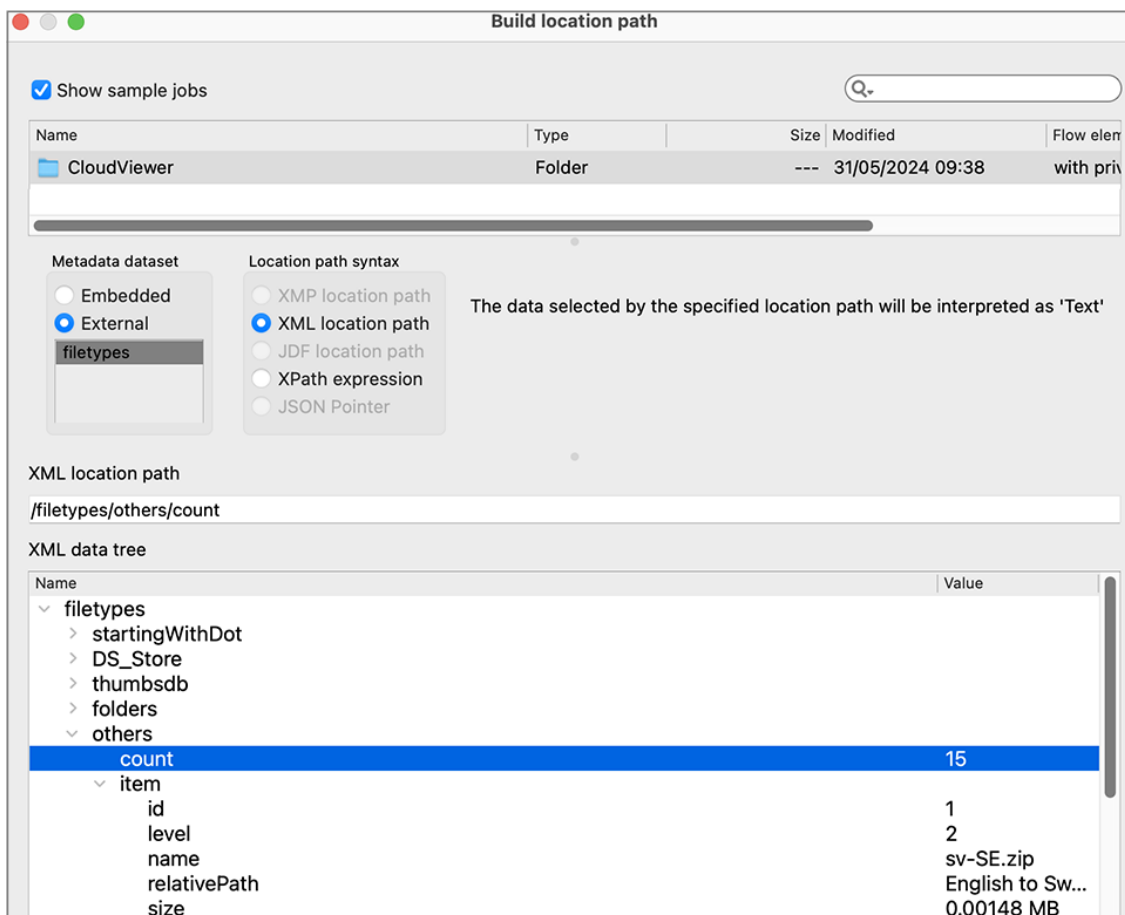
Example of private data (private data key prefix is filetypes in this example):
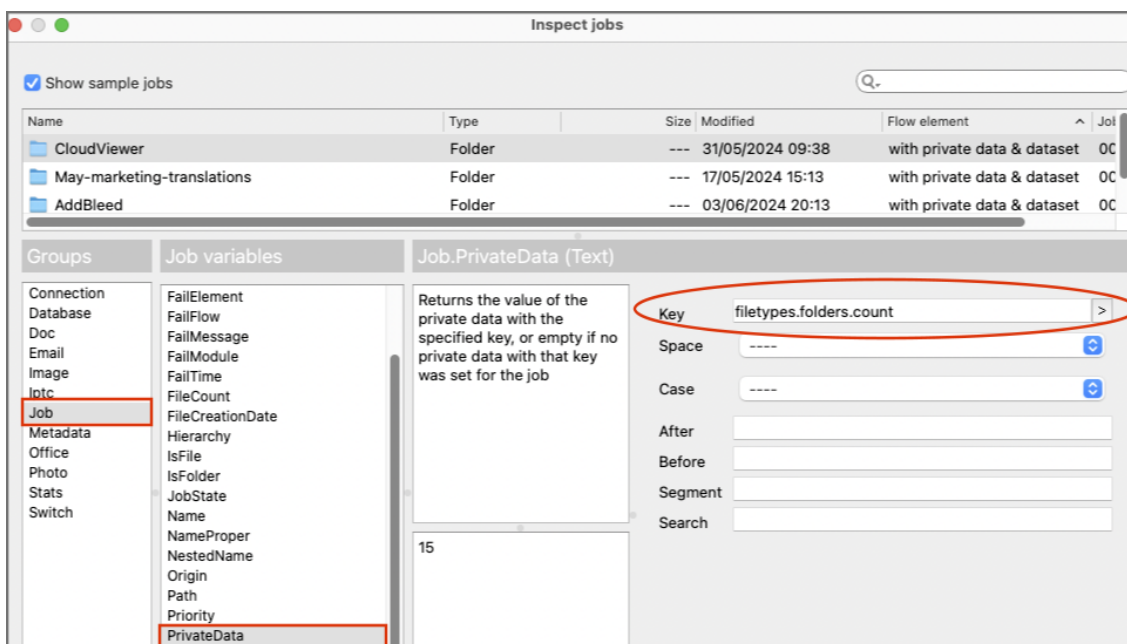
Delete files

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.

Delete files 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 and the property to allow deleting all files is set to "No".

In all other cases, the job folder is sent along the success connection, also when no files have been deleted.

Delete files Properties:

Properties	Description
Delete or keep files	This property determines whether the files matched by the properties below will be deleted or kept. Options: • Delete (default) • Keep
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.
Allow deleting all files	When set to "No" (default) the job folder will be sent to the Error connection should the setup be such that ALL files would be deleted from the job folder. Setting the value to "Yes" allows the folder to become empty and it will be sent along the Success connection.
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.
Remove empty folders	Determines whether or not empty folders will be removed. Options are: • Yes: The empty folders will be removed recursively, i.e. folders that contain empty subfolders will also be removed. • No (default): The empty folders won't be removed.

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 (*hotfoldermonitor.html*) 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. (To do so, in the properties of Mail send, set *Include attached body text* to Yes. To include hotfoldermonitor.html as attachment, set *Attach files* to Yes.)

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 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.
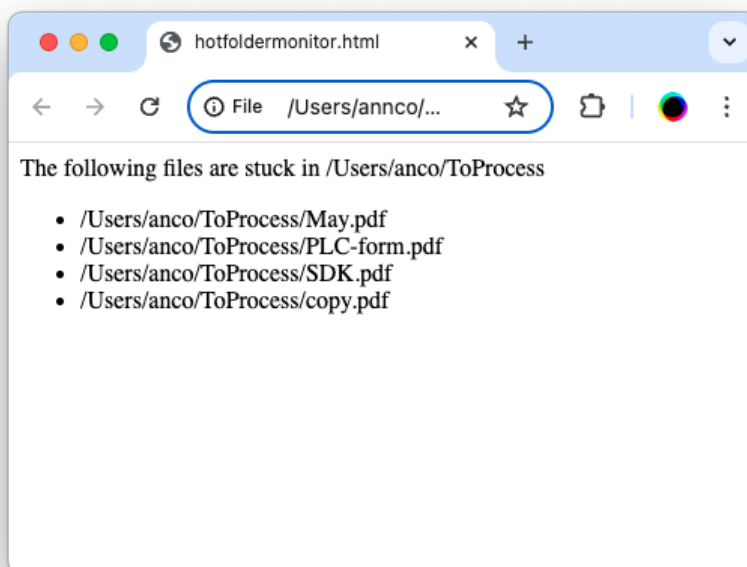
Hot Folder Monitor Connections:

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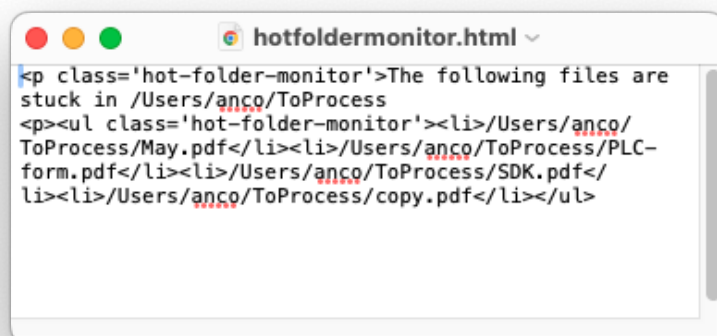
Hot Folder Monitor Properties:

Properties	Description
Hot folder to be monitored	The folder to be monitored. There is no default. It is also possible to use a string to define the path. If the path does not exist (it could be removed as the flow is running), the element will go into a fail state and a notification will be sent if that is set up in the Preferences.
Interval (secs)	Determines how often the folder is scanned for changes. Default: 300 **Note:** • Make sure to choose a value that is higher than the value used by the application that scans the monitored folder. • Allow for some extra processing time to reduce the chance of false alarms. Default: 300.

Properties	Description
	• When the application is not running for a longer period, the app will trigger the notification every time it runs!
Exclude file types	Ignore certain file types. Default value is None.
Exclude file patterns	Ignore certain files based on the defined patterns. Default value is None.
Exclude regular expressions	Ignore certain files based on regular expressions. Default value is None.

hotfoldermonitor.html (example):

hotfoldermonitor.html (example - HTML source):

```
<p class='hot-folder-monitor'>The following files are
stuck in /Users/anco/ToProcess
<p><ul class='hot-folder-monitor'><li>/Users/anco/
ToProcess/May.pdf</li><li>/Users/anco/ToProcess/PLC-
form.pdf</li><li>/Users/anco/ToProcess/SDK.pdf</
li><li>/Users/anco/ToProcess/copy.pdf</li></ul>
```

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 can be removed from their location and injected into the flow for further processing, e.g. archiving them in a different location, or they can be deleted in situ. One use case for this tool is to clean up the location where you store customer files 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 ("files_not_removed_from_the_file_pool.html") 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.

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 because of the cleanup are not immediately removed, but they will be the next time.

The app can inject the files as new jobs into the flow and delete the original file, or it can also just delete the file without injecting anything. If the strategy is to physically remove the files, the latter choice is better (see further in the properties).

The app can also create a log file, both in the case of success and in the case of errors. The user can define a couple of error conditions (see further in the properties) but deleting a file can also fail when Switch does not have sufficient access rights. In both cases a very simple HTML file is created. 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: a class is defined for the p, ul and li tags (<p class='file-poolcleanup-p'>, <ul class='file-pool-cleanup-ul'>, <li class='file-pool-cleanup-li'>. This allows you to define those classes in the CSS of the HTML template used in "Mail send" and in this way you can format the text to your liking.

The app can set the hierarchy path of the injected job.

File pool cleanup Connections:

File pool cleanup can be used with an incoming connection or without it. Without it, the cleanup is triggered at regular time intervals as defined in the properties; with it, the cleanup is triggered by an incoming job.

The app has outgoing traffic-light connections, a Data Success one for the files that have been cleaned up from the file pool, a Log Success connection with a logfile that lists all the files that have been removed from the file pool, and a Log Error one to which an error log file is sent, e.g. when a limit has been exceeded or a file could not be deleted. The names of the log files depend on the error and they are self-explanatory.

File pool cleanup Properties:

Property	Description
Root folder	Root folder to be cleaned up. There is no default.
	Subordinate property: *Allow root and network folders*:
	• Yes: Root and network folders are allowed. This property can also be defined through single-line text with variables.
	• 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!**
Start time	Start time of the check. Options:
	• Now (default): When the flow starts, cleaning is started immediately and repeated as defined in the Scan interval property.
	• Custom: A *time of day* can be defined at which the first cleaning will start. When this time is earlier than the moment at which the flow is started, the first cleaning will only take place the next day. It is not mandatory, but it is logical to combine this choice with an interval of 1 day. The app checks every 5 minutes if the current time is later than the expected next run

Property	Description
	time. This means that the actual run time can be a couple of minutes later than the defined one.
Scan interval	Time between two checks. Default: 1 Subordinate property: *Unit of time*: There is a choice of Days (default), Hours and Minutes. When choosing minutes, it is not possible to define an interval value of less than 10.
Remove files older than	Time after which files are removed. Default: 30. *File date type:* Options are: • Modification date (default) • Creation date *Unit of time:* Options are: • Days (default) • Hours • Minutes
Remove empty folders	Determines whether or not empty folders are removed. Options are: • Yes (default) • No **Note:** Folders that become empty because of the cleanup are not immediately removed, but they will be the next time.
Ignore files	Determines which files should never be cleaned. You can use the File types and File patterns editors for defining file specifications. Default: None.
Ignore folders	Determines which folders should never be cleaned. You can use the Folder patterns editors for defining folder specifications. 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. **Tip:** During the setup phase, set this property to 1 to get a log file, so you can see if the correct files are removed! Do not forget to configure a Log error connection.

Property	Description
Delete method	Options:
	• Inject (default): The file is injected into the flow and deleted from the file pool. This is interesting if the files must be archived somewhere else. This option comes with two extra properties:
	• *Maximum batch file size (GB)*: When injecting jobs into the flow, they take up disk space where the Switch application data root is stored. If the file pool is on a disk that is much larger than the application data root disk, it is theoretically possible that the cleaning process fills up the disk bringing Switch to a halt. If the combined file size of the files to be removed is larger than this value, nothing is deleted. If you are in such a case, it is advised to use the Delete option (see below) or to use multiple instances of File Pool Cleanup that scan smaller parts of the file pool. Default: 1.
	• *Attach hierarchy info*: When set to "Yes" the hierarchy path to the file that was cleaned up will be added to the job ticket.
	• Delete: The file is not injected in the flow, but deleted from the file pool. If the cleaned-up files do not have to be archived, but only physically removed, this option is faster and does not have the risk of filling up the application data root disk.

File Sorter and Spacer

Unsorted   File Property Sort   Sorted

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.

File Sorter and Spacer Connections:

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

File Sorter and Spacer Properties:

Property	Description
Sort property	Property to sort the files on. Options are: • File creation date • File modification date (default) • File size • File name • Number of pages • Page surface (only for PDF and image files)
Sort order	Sort order. Options are: • Ascending (default) • Descending
Space files in job folder	Options are: • No (default) • Job folder after job folder: All files of all waiting job folders trickle through one file at a time • Multiple job folders at the same time: There will be one file from every waiting job folder that is injected as a new job at every space interval If enabled, the following properties are triggered: • *Space interval (secs)*: 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. Contrary to Submit hierarchy this app does not allow a level selection, it always searches to the bottom of the hierarchy.

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 allows to leave the original files in place without being injected a second time and it 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.

Submit Filespec Connections:

Submit Filespec does not have any incoming connections.

The outgoing connections are Success and Error connections.

Submit Filespec Properties:

Property	Description
Path	Root folder of the hierarchy to be scanned. There is no default.
	Subordinate properties:
	• *Allow root and network folders*:
	• Yes: Root and network folders are allowed.
	• 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!**
	• *Local copy mode*: Especially when using network folders, there's a risk that because of missing access rights, a file cannot be deleted after it has been injected as a job.
	When set to Yes, the app will first make a copy of the file to the local computer, try and delete the remote file and depending on whether this was successful or not, send the new job to the Success or Error connection.
Scan interval (secs)	Time between two checks. Default: 30
File types	Filetypes that have to be submitted into the flow. Default: None

Property	Description
File patterns	File patterns of files that have to be submitted into the flow. You can use 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	Regular expressions to define the files that have to be submitted into the flow. You can use wildcards, ? standing for 1 unknown character, and * for any number of unknown characters. There can be multiple regular expressions, one per line. Default: None.
Remove empty folders	Determines whether or not empty folders are removed. Options are: • Yes (default) • No **Note:** 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.
Leave originals	Determines whether or not originals are kept: • Yes: The files are injected into the flow, but they are left in the original location and will not be injected again during later iterations, also when the flow has been stopped and started in the meantime, and also when the file is overwritten. This option triggers an extra property: *Ignore updates*. • No (default)
Attach hierarchy info	Determines whether or not the hierarchy path to the file that was injected will be added to the job ticket: • Yes (default) • No