

Wait For Asset +



V1.0 Documentation



Overview

An enhanced version of Wait For Asset which intelligently monitors, processes, and manages digital assets within automated workflows, providing robust file handling capabilities with advanced pattern matching, conditional processing, and flexible asset management options.

Hold a job in a flow until a path for a file or folder has been found. If the timeout has been reached, the incoming job will be moved down the error traffic light path. There is also the capability of injecting the asset found, and/or attaching a dataset of the asset found when conditions are met.

Compatibility

Switch 2024 Fall and higher. Windows and Mac OSX

Incoming Connections

At least one incoming connection required.

Outgoing Connections

At least one outgoing Traffic Light connection is required.

Flow Element Properties

- Asset (1-10): An absolute path or [wildcard](#) path to a file or folder to check for.
 - Mode: Applicable when wildcard patterns are utilized in the above path. Wait for Asset can only return one file or folder per group. (See [Selection Modes](#))
 - Inject: After an asset is found/exists, do we want to push this forward – in place of the Incoming Job? (Yes/No).
 - Name: What is the name of the outgoing injected job?
 - **Automatic**: The current name of the incoming job.
 - **Current**: The current name of the injection job.
 - Type: Will the app push out a child job (including all relevant data from the parent job), or a new job. This property is ignored if the Settings property "Group in folder" is "yes".
 - Add as dataset: Specify whether to attach the searched asset as an outgoing dataset when found (Yes/No).
 - Name: The name of the newly created dataset.
 - Type: The dataset type specific to the asset.
 - Delete: Will this delete the file or folder found after injection? (Yes/No)
- Settings
 - Condition met: How does the app handle finding multiple assets? Do we wait for all assets to be found before moving forward? Or do we find the first found and move forward. (All Found/First Found)

- Success on timeout: If the timeout has been reached, does the app push any "found" assets to the Success connection path despite not meeting its condition. (Yes/No)
- Group in folder: Group all found assets into a single folder? This property is only applicable when injecting one or more assets. (Yes/No)
 - Zip folder: Setting this property to "yes" will zip the contents as a compressed zip file on output. (Yes/No)
- Interval: The frequency in which we attempt to validate the asset location (Seconds).
 - Units: Type of unit for the interval determination. (Seconds/Minutes/Hours/Days)
- Timeout: The amount of time to wait before timing out and moving down the failure path.
 - Units: Type of unit for the interval determination. (Seconds/Minutes/Hours/Days)

Wildcard Patterns

Wait For Asset + supports powerful wildcard patterns (also known as "glob patterns") for flexible asset matching. Instead of specifying exact file paths, you can use wildcard patterns to match multiple files and automatically select the most appropriate one based on your criteria.

Why Use Wildcards?

Wildcards are essential when:

- File names change dynamically (timestamps, version numbers, etc.)
- You need the most recent file from a directory
- Processing batches where exact names are unknown
- Handling files generated by external systems with variable naming

Supported Wildcard Patterns

Basic Patterns

Pattern	Description	Example	Matches
*	Any # of characters (except path separators)	report_*.pdf	report_2025.pdf, report_final.pdf
?	Matches exactly one character	data_?.xml	data_1.xml, data_A.xml
[abc]	Matches any single character in brackets	file_[123].txt	file_1.txt, file_2.txt, file_3.txt
[a-z]	Matches any character in range	report_[A-Z].pdf	report_A.pdf, report_Q.pdf
[!abc]	Matches any character NOT in brackets	data_[!0].csv	data_1.csv, data_A.csv (not data_0.csv)

Advanced Patterns

Pattern	Description	Example	Matches
**	Recursive directory matching	logs/**/error.log	logs/error.log, logs/2025/error.log, logs/app/debug/error.log
{a,b,c}	Matches any of the alternatives	{art,proof}_*.pdf	art_123.pdf, proof_456.pdf
!(pattern)	Negative matching (extended glob)	!(backup_*)	Any file not starting with backup_

Selection Modes

When a wildcard pattern matches multiple files, use these modes to select which one to process:

Available Modes

Mode	Description	Use Case
first	First file alphabetically (<i>default</i>)	Consistent processing order
last	Last file alphabetically	Most recent by name
newest / latest	Most recently modified file	Latest data processing
oldest	Oldest modified file	Historical data first
largest	Largest file by size	Complete datasets
smallest	Smallest file by size	Summary reports

Cross-Platform Compatibility

The system automatically handles path differences between operating systems:

Input	Normalized	Platform
C:\data*.csv	C:/data/*.csv	Windows
/home/user/*.txt	/home/user/*.txt	Linux/Mac

Common Use Cases

1. Processing Latest Reports

- **Pattern:** reports/daily_report_*.xlsx
- **Mode:** newest
- **Result:** Always processes the most recent daily report

2. Batch Invoice Processing

- **Pattern:** invoices/pending/*.pdf
- **Mode:** first
- **Result:** Processes invoices in consistent alphabetical order

3. Log File Analysis

- **Pattern:** logs/**/application.log
- **Mode:** largest
- **Result:** Finds the most complete log file across all subdirectories

4. Backup File Selection



- **Pattern:** backups/database_backup_*.sql
- **Mode:** latest
- **Result:** Always uses the most recent database backup

5. Template Matching

- **Pattern:** templates/{invoice,receipt,statement}_template.{pdf,docx}
- **Mode:** first
- **Result:** Matches any template type in multiple formats

Best Practices

1. Be Specific

-  *.* (too broad, may match system files)
-  reports/*.xlsx (specific file type and location)

2. Use Appropriate Modes

- Use newest for time-sensitive data
- Use first for consistent processing order
- Use largest when file size indicates completeness

3. Test Your Patterns

Start with simple patterns and gradually add complexity:

Step 1: data/*.csv

Step 2: data/sales_*.csv

Step 3: data/sales_*_[0-9][0-9][0-9][0-9].csv

Troubleshooting

No Matches Found

- **Check case sensitivity:** Data/*.CSV vs data/*.csv
- **Verify path separators:** Use forward slashes / consistently
- **Test incrementally:** Start with * then add specificity

Too Many Matches

- **Add specificity:** report_*.xlsx → report_daily_*.xlsx
- **Use date patterns:** *_2025-*.pdf for current year only
- **Leverage subdirectories:** current/*.csv vs archive/*.csv

Unexpected Selection

- **Review mode choice:** newest vs largest can yield different results
- **Check file timestamps:** System clocks or file transfers may affect newest
- **Verify sorting:** first/last use alphabetical order, not date order

Performance Considerations

- **Limit search scope:** data/*.csv is faster than **/*.csv
- **Avoid overly broad patterns:** *.txt searches entire directory tree
- **Use specific extensions:** *.log vs * when you know the file type

About Significans Automation



Significans Automation is a software integrator specializing in delivering next-generation automation to the Printing and Packaging industry.

We offer programming and expertise in custom workflow development, deployment of communication and project management systems, color management, and end-to-end business integration. While upholding software neutrality, Significans Automation advises and tailors best in class software to optimally fit the environment.

The level of sophistication that is provided increases profitability, improved quality control, and enhanced production efficiency, enabling Artificial Intelligence and Robotics, while also facilitating new revenue opportunities in e-commerce. We are driven by the conviction that customized automation is the only path forward.