

# FORTRA

Automate  
24.1.0

## Best Practices and Optimizations Guide

## **Copyright Terms and Conditions**

---

Copyright © Fortra, LLC and its group of companies. All trademarks and registered trademarks are the property of their respective owners.

The content in this document is protected by the Copyright Laws of the United States of America and other countries worldwide. The unauthorized use and/or duplication of this material without express and written permission from Fortra is strictly prohibited. Excerpts and links may be used, provided that full and clear credit is given to Fortra with appropriate and specific direction to the original content.

202403040550

# Table of Contents

|  |           |
|--|-----------|
| <b>Overview</b>  | <b>1</b>  |
| <b>Performance</b>                                       | <b>2</b>  |
| Triggers   | 2         |
| Workflow design  | 3         |
| Task Design  | 4         |
| General/other considerations                             | 4         |
| <b>Security</b>  | <b>6</b>  |
| Secure the Automate environment                          | 6         |
| Secure the Automate installation                         | 6         |
| Secure Automate user access                              | 7         |
| Secure sensitive data                                    | 7         |
| <b>Development</b>                                       | <b>9</b>  |
| The development environment                              | 9         |
| Workflows  | 9         |
| Tasks  | 10        |
| Conditions   | 12        |
| Data containers (Variables, Datasets, Constants, Arrays) | 12        |
| General development                                      | 13        |
| <b>Optimizations</b>                                     | <b>15</b> |
| Automate installation architecture                       | 15        |
| How Automate runs scheduled jobs                         | 16        |

How to detect and avoid excessive triggering ..... 18

Optimum workflow designs .....22

Additional optimization settings .....25

# Overview

This guide is designed to provide best practices related to performance, security, development, and optimizations for Automate, based on Fortra's experience working with Automate customers. These best practices are provided to promote long-term maintenance of your robotic process automation (RPA) project. While this document provides recommendations, we recommend that each organization create and maintain their own internal processes and best practices.

Think of best practices as an art as they involve competing requirements. For example, maintainability may, in some cases, compete with performance. As such, the content provided here is intended to be a guide rather than a set of strict rules.

The most common and avoidable issues Automate customers encounter are related to performance, specifically using Automate in ways that unnecessarily burden the application. Therefore, a major portion of this document deals with best practices related to performance.

We also understand that some of our customers are as experienced as we are, if not more. We invite you to join our [Automation Insiders](#) program to share your thoughts and suggestions. Based on your feedback, we will review discussions and update this guide periodically.

In addition, we offer the Automate Assessment performance tuning service <https://www.fortra.com/resources/guides/helpsystems-automate-services-guide#assessment-services>. You can request this service at <https://www.fortra.com/product-lines/automate/services-and-trainings/request>. However, this guide captures most of that knowledge to help you get the most of your Automate application.

# Performance

Many Automate customers tend to develop their tasks and workflows in the simplest, most straightforward way, without giving any thought to security, performance, or long-term maintenance of the system. Initially, this does not cause any issues, but it tends to become a problem over the course of months or years after adding many tasks and workflows to the Automate application.

The most common adverse effects to Automate performance are:

- Execution server CPU usage spikes around the time workflows are scheduled to start, slowing down the system overall.
- Orphaned workflows or tasks. This occurs when:
  - A task or workflow's status it has been running for extended periods of time.
  - A workflow's status shows it is in Running mode with no tasks.
- Delayed task and workflow execution.
- Memory utilization building up on the Execution Server or agent machines.
- Excessive growth in the Execution Logs table in the back-end database.
- The Execution Output panel shows multiple or a growing number of workflows or tasks in Pending status.
- Agents are reporting they are offline.
- Execution Server or agent system application event logs are flooded with repetitive error messages.
- System failure at the server or with agents.

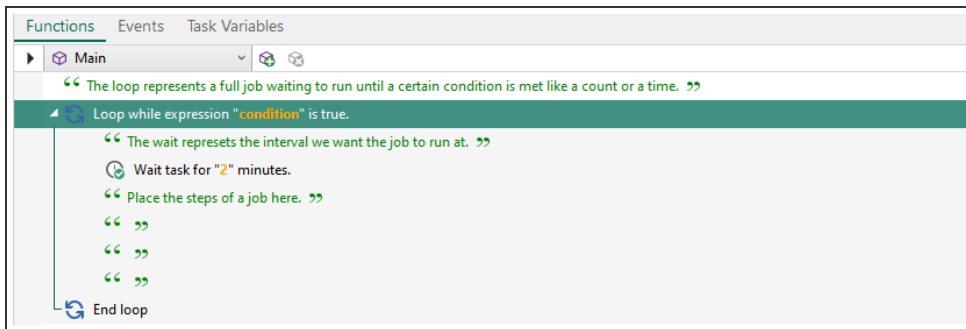
The presentation of symptoms may differ, but the cause is the same in all cases – there are costs associated with automation that are not accounted for. For each workflow, the trigger requirements must be met and recognized or fired, automation markup language (AML) must be interpreted and transmitted across the network, and variables and structures must be instantiated to do the work. These operations happen quickly, but they are not free and enough of them triggered together can overwhelm even the most well-provisioned system.

## Triggers

### Stop triggering tasks or workflows so aggressively

Triggers are expensive regarding resource consumption. As such, be mindful when using triggers, especially schedule-based triggers which fire in short intervals. If the work

associated with a schedule-based trigger cannot be completed before the next firing of the trigger, the trigger can cause an increasing workload that can eventually overwhelm the system. Alternatively use in-task loops to replace iterative tasks.



## Balance the triggering

Rather than setting up many triggers to fire at the same time or using the same schedule trigger for many tasks and workflows, which can result in a performance spike, try to distribute the triggers over a period.

## Reduce the number of file triggers

Monitoring file systems is an expensive in terms of maintenance. Use file triggers for small amounts of infrequent file changes. If you are expecting an enormous number of files in a short period of time, switch to a schedule-based trigger and batch process the workload or use a loop to monitor the file system. Using a loop ensures that the file system is queried only after the current work on it is complete.

In addition, avoid monitoring network shares or replace them with a local agent when possible. Generally, try to avoid placing multiple triggers within same workflow.

## Workflow design

### Avoid multiple independent branches in a single workflow

We strongly recommend to not use multiple independent branches in a single workflow as it makes it difficult to trace individual branch execution from the execution logs. The workflow engine is not optimized for this kind of structure, and it can cause performance issues. Instead, use different workflows for each independent branch.

### Use Previous Agent and Trigger Agent

When possible, use the Previous Agent and Trigger Agent System Agents in your workflow. You can pair the Trigger Agent with Previous Agent for even more flexibility. Avoid Static Agent assignment whenever possible.

## Task Design

### Set task priorities correctly

Make use of task priorities to prioritize your workload accordingly and to achieve the best results. We recommend setting tasks to fail if their Priority condition threshold is exceeded to track and audit the execution logs. Refrain from keeping the default settings of Run on ALL Agents. Round Robin is a good option to distribute the load among all agents. See *Priorities* in the Automate User Guide on the Fortra Support Portal at <https://support.fortra.com/> for more information.

#### Task Priority

Specifies whether the task must run exclusively of other tasks and how to handle situations where another task is already running.

☐ Always run task  
☒ Run task if the number of running instances of this task is below the threshold

Running instances threshold:

1

If the condition above is not met:

☐ Hold task until condition is met  
☐ Hold task, then abort task if still not met  
☐ Hold task, then interrupt all running tasks and run task  
☐ Interrupt all running tasks and run task  
☐ Interrupt all running instances of this task and run task  
☒ Do not run task

If the task does not run, treat as:

☐ Success  
☒ Failure

☐ Run task if the number of running instances of all tasks is below the threshold  
☐ Run task if no other tasks are running on the agent

## General/other considerations

### Use a dedicated machine for the Execution Server and Management Server

We recommend you use the following deployment architecture:

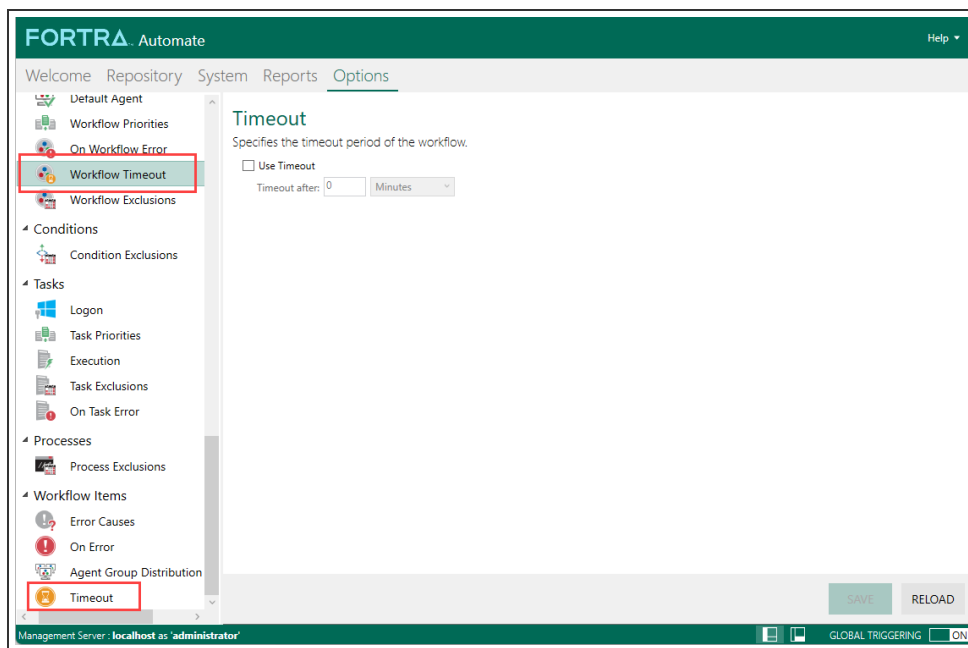
- Use a dedicated machine for the Execution Server and Management Server
  - Do not install an agent on this machine
  - Do not run studio tools on this machine



- Use an appropriate datastore if you intend to use the system with a large number of workflows/tasks
  - It is not recommended to use free versions of DB engines (for example, preinstalled SQL Express or Oracle) as they are typically not fully functional and key features are not accessible until a full version is purchased.
  - If performance is suffering, we recommend moving the datastore to its own dedicated machine

## Correctly set Workflow and Task Timeouts

Setting valid Workflow Timeout and Task Timeout helps track stalling tasks and resources locks. Some of the main causes for overloaded systems is tasks hanging on the agent side, which can increase memory usage and cause CPU spikes which results in queued tasks and delays with triggering and execution. See *Workflow Timeout* and *Timeout* in the Automate User Guide on the Fortra Support Portal at <https://support.fortra.com/> for more information.



# Security

Security recommendations and best practices include securing the Automate environment, installation, and user/group access.

## Secure the Automate environment

The following are guidelines for securing the Automate environment:

1. Secure your physical computer infrastructure using industry standard practices. See the following two resources for more information:
  - a. The Center for Internet Security website <https://www.cisecurity.org/>.
  - b. The University of Connecticut's Server Hardening Standard (Windows) guide <https://security.uconn.edu/server-hardening-standard-windows/#>.
2. Secure the Windows operating system using industry standard practices. See the NIST SP 800-123, Guide to General Server Security document at <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-123.pdf> for more information.
3. Secure the database using industry standard practices. See the following Microsoft documents for more information:
  - a. Securing SQL Server <https://docs.microsoft.com/en-us/sql/relational-databases/security/securing-sql-server?view=sql-server-2017>.
  - b. See SQL Server 2016 STIG YY22M01 Checklist Details <https://nvd.nist.gov/ncp/checklist/838>.

## Secure the Automate installation

The following are guidelines for securing the Automate installation:

1. Configure a secure connection to the database. See Microsoft's Enable encrypted connections to the Database Engine doc at <https://docs.microsoft.com/en-us/sql/database-engine/configure-windows/enable-encrypted-connections-to-the-database-engine?view=sql-server-2017> doc for more information.
2. Configure SSL/TLS connections between components.
3. Configure Server Management Console (SMC) Sessions in Automate. See *SMC Sessions* in the Automate User Guide on the Fortra Support Portal at <https://support.fortra.com/> for more information.

4. Set the Administrator's password in Automate. See *Users* in the Automate User Guide on the Fortra Support Portal at <https://support.fortra.com/> for more information.
5. Enable API access in Automate. See *API Security* in the Automate User Guide on the Fortra Support Portal at <https://support.fortra.com/> for more information.
6. Configure the Event Monitor to prevent user interaction by way of the following parameters in Automate:
  - a. From the Automate Server Management Console's navigation bar, select **Options > Default Properties > Indicators**.
    - i. Select **Enable right-click menu on tray icon**.
    - ii. Select **Enable the task interruption hotkey**.
    - iii. In the **Task interruption hotkey** box, enter the desired hot key combination.
  - b. From the Automate Server Management Console's navigation bar, select **Options > Default Properties > Execution**.
    - a. Select **User can stop this task**.

## Secure Automate user access

The following are guidelines for securing Automate user access:

1. Define user groups/roles and permissions. To streamline user access management, create user groups, assign users to those groups, and then assign permissions to the roles. See *User Groups* in the Automate User Guide on the Fortra Support Portal at <https://support.fortra.com/> for more information.
2. Apply item permissions throughout product, by configuring which users have permissions to each object. See *System Security (Permissions)* in the Automate User Guide on the Fortra Support Portal at <https://support.fortra.com/> for more information.
3. Apply system-wide permissions. Configure which permissions user groups and/or users have in Automate. See *System Security (Permissions)* in the Automate User Guide on the Fortra Support Portal at <https://support.fortra.com/> for more information.

## Secure sensitive data

Follow secure development practices when writing workflows & tasks by configuring the following options in Automate:

1. Create credential variables to share among all or specific users and/or groups. See *Credentials* in the Automate User Guide on the Fortra Support Portal at <https://support.fortra.com/> for more information.
2. Create constants on the agent machine. See *Constants* in the [Automate User Guide](#) on the Automate User Guide on the Fortra Support Portal at <https://support.fortra.com/> for more information.
3. Create a SQL connection for each agent. See *SQL Connections* in the Automate User Guide on the Fortra Support Portal at <https://support.fortra.com/> for more information.

# Development

Below are the best practices for maintaining a healthy development environment.

## The development environment

Document all development environments, especially if overall ecosystems contain multiple environments (for example, PROD, DEV, etc.). Define what each environment is used for, as well as the number of development tools and agents for each environment.

## Using Agent Groups

Set the appropriate Agent Group distribution when using them. Create a user group for each agent, assign the agent to that group, and then assign items in workflows to the group. This set up makes it easy to swap one agent for another one and is useful for when you replace an agent with new hardware, or you need to take it offline for maintenance.

See *User Groups* and *Agent Groups* in the Automate User Guide on the Fortra Support Portal at <https://support.fortra.com/> for more information.

## Workflows

For general information on Workflows, see *Workflows* in the Automate User Guide on the Fortra Support Portal at <https://support.fortra.com/> for more information.

## Workflow organization and nomenclature

Organize workflows in a way that best suits your needs, whether by Application, Line of Business, Department, etc. Begin with an agreed upon foundation and create and organize Automate folder structures at the Workflows and Repository levels. For example, if you organize workflows by department, create a Human Resources workflow.

When naming workflows, keep names consistent with the organizational structure put in place at the workflow level:

- Keep workflow names under 25 characters.
- Consider using a standard prefix.
- Decide on a workflow name delimiter (for example, "\_" or "-").

- Use intuitive and meaningful names. For example, while creating workflows in the Human Resources folder, prefix each workflow name with "HR" followed by a workflow name that describes the process (for example, "HR Account Creation").

## Workflow development

You can simplify workflow creation by using a standard nomenclature for data containers, tasks, and making use of the Automate Repository. Dynamic task, condition, and data container creation aids in this area. See *Tasks, Events & Conditions*, and sections on Data Containers (*Variables, Datasets, Arrays, and Constants*) in the Automate Plus/Ultimate User Guide on the Fortra Support Portal at <https://support.fortra.com/> for more information.

When creating workflows, create a roadmap by writing out your workflow process including triggering events, conditions, logic, and tasks on paper or a whiteboard before building them in Automate. This approach allows you to create a clear view of automation requirements, as well as help save time with formatting your task or workflow.

Use a workflow's **Notes** property to document important information regarding it, allowing users to understand the reasoning behind all objects, concepts, and logic used with the workflow. To access the **Notes** property:

1. Double-click your workflow.
2. Select **Notes**.
3. Enter a note in the box.
4. Select **OK**.

## Workflow security

Set group and user permissions for each workflow. To set permissions:

1. Double-click your workflow.
2. Select **Security**.
3. See *System Security (Permissions)* in the Automate User Guide on the Fortra Support Portal at <https://support.fortra.com/> for more information.

## Tasks

For general information on Tasks, see *Tasks* in the Automate User Guide on the Fortra Support Portal at <https://support.fortra.com/> for more information.

## Task organization and nomenclature

Task organization should follow the same general guidelines as workflows:

- Keep task names under 30 characters.
- Consider using a standard prefix.
- Decide on a task name delimiter (for example, "\_" or "-").
- Use intuitive and meaningful names.

## Task development

- Modularize tasks for reusability. Small, dynamic tasks make for quick and seamless workflow development by way of the Repository.
- Stick to a standardized naming convention for data containers so that you can plug tasks into any workflow or function.
- Use a task's Notes property to document essential information regarding it, allowing users to understand the reasoning behind all objects, concepts, and logic used with the task.

To add a note:

1. Double-click your workflow.
  2. Select **Notes**.
  3. Enter a note in the box.
  4. Select **OK**.
- Build and test tasks in the Automate 2024 Task Builder Debug Panel first. Once a task runs correctly, run the task from the Workflow Designer to ensure that it has the correct configuration, credentials, and settings to run on the agent (that is, Bot) properly. See *Task Builder Debug Panel* and *Workflow Designer* in the Automate User Guide on the Fortra Support Portal at <https://support.fortra.com/> for more information.
  - Follow these guidelines when building and testing in the Task Builder:
    - Set Breakpoints. See *Breakpoints* in the Automate User Guide on the Fortra Support Portal at <https://support.fortra.com/> for more information.
    - Create Regions within larger tasks to make the task steps easier to follow. See *Regions* in the Automate User Guide on the Fortra Support Portal at <https://support.fortra.com/> for more information.
    - Save your task frequently.
  - When using Sessions, always include a Close Session step with the task to end that session.

- For password fields, use Credentials whenever possible. See *Credentials* in the Automate User Guide on the Fortra Support Portal at <https://support.fortra.com/> for more information.
- For file paths, use the Constants option. See *Constants* in the Automate User Guide on the Fortra Support Portal at <https://support.fortra.com/> for more information.
- For any kind of stand-alone number, place it in a descriptive variable or constant.

## Task security

Set group and user permissions for each task. To set permissions:

1. Double-click your task.
2. Select **Security**.
3. Select a Group or User Name from the **Available** box, and then select **Move** to move it to the **Selected** box.
4. Select **OK**.

See *System Security (Permissions)* in the Automate User Guide on the Fortra Support Portal at <https://support.fortra.com/> for more information.

## Conditions

Automate supports numerous condition types such as File System, Database, and Schedule Conditions (Triggers).

## Conditions nomenclature

When naming conditions, follow these guidelines:

- Keep condition names under 30 characters.
- Use a prefix to distinguish conditions. For example, “SCH” for Schedule Condition, “DB” for Database Condition. You can create subfolders for better organization.
- Use a standard naming based on interval and trigger time. For example, using delimiters such as “\_”, for schedule trigger running every 20 minutes SCH\_20M , for schedule trigger running daily at 7 AM : SCH\_Daily\_7AM. Add a reference to the job as well to help track the execution like: SCH\_FTP\_xxx\_Daily\_7AM.

## Data containers (Variables, Datasets, Constants, Arrays)



- Use a standard prefix for each of the different types of containers.
- Use intuitive and meaningful names.
- Document the agreed-upon nomenclature and follow the chosen conventions across all workflow and task development.
- Only use the underscore character as a delimiter (\_), or nothing at all:
  - Variables
    - Prefix variables with **var** or **VAR**:
      - varDate
      - VAR\_Username
  - Datasets
    - Prefix datasets with **ds** or **DS**:
      - dsTest
      - DS\_Results
  - Constants
    - Prefix constants with **cst** or **CST**:
      - cstFilename
      - CST\_FolderPath
  - Arrays
    - Prefix arrays with **arr** and **ARR**:
      - arrData
      - ARR\_Data

## General development

- Remember “The last saved window rule”:
  - Ensure that only one instance of the Workflow Designer and/or Task Builder are open on your desktop at any given time.
    - If multiple instances of the same Task are open in the Task Builder, the last window where changes are saved will override any other instances of that task.
    - If multiple instances of the same Workflow are open in the Workflow Designer, the last window where changes are saved will override any other instances of that workflow.
- Use documentation generously. This produces a clear picture of all the intricate details affecting the components of the Automate environment and captures the intent of the design.

- When using loops, build and test the automation process outside of the loop to ensure automation behaves as expected, and then move those steps within the loop to continue building/testing.
- Use correct formatting and indentions. For example, if looping, indent all the steps/objects within the loop to make the task easier to read, understand, and follow.
- If automating FTP or SharePoint processes, utilize the built-in browsers within the Action Configuration properties.
- Create a standard error handling procedure for all Automate developers to follow, if possible.
- Append dates and times to files and logs whenever possible.
- Use global options like Constants, SQL Connections, Global Email Settings, Default Agents, and Default User whenever possible. If multiple tasks use the same Constant or SQL Connection, you can perform changes in one place rather than opening each task associated with the changed resource.
- The File Systems monitoring (trigger/condition) only applies to Windows. It cannot monitor FTP folders.

# Optimizations

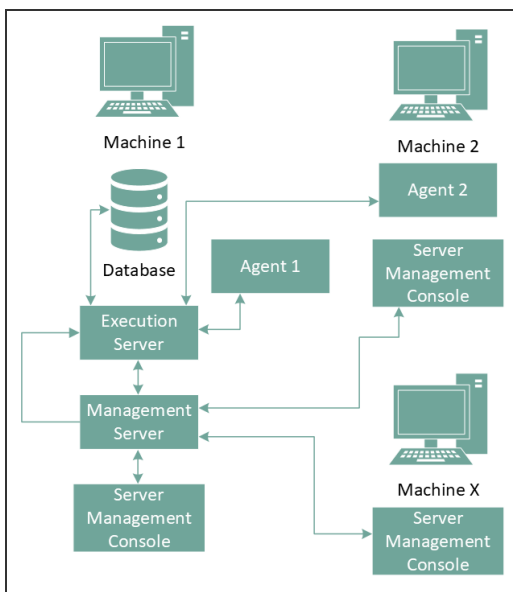
How you design Automate jobs and configure your environment dictates the performance and stability of the system. As your organization grows, you will add more jobs over time. Most Automate developers tend to use repetitive approaches in workflow designs, and reuse triggers and tasks to minimize development and updates.

The purpose of this chapter is help Automate developers and administrators optimize Automate jobs to use minimal system resources.

## Automate installation architecture

Installation architecture is important for overall system efficiency. For example, a simple environment setup of Automate would include two agents and three Server Management Console installations (each installed using default options). The following design diagram illustrates this environment and shows which modules are installed on each machine:

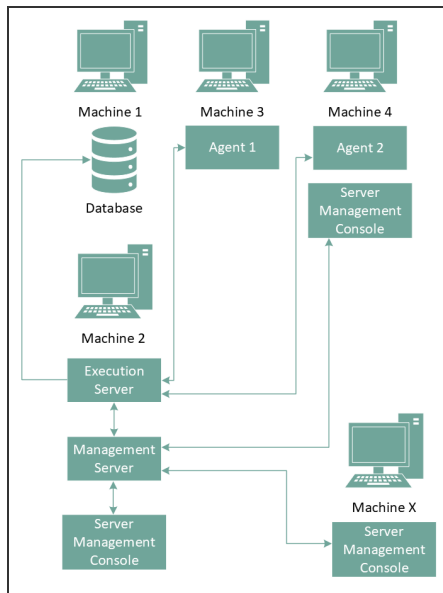
- Machine 1 - Execution Server, Management Server, SQL Server Express Database, Agent 1, and Server Management Console
- Machine 2 - Agent 2 and Server Management Console
- Machine X - Server Management Console



In Automate, the Execution Server, database, and agent machine modules consume the most resources in a functional environment. To improve the design in the diagram, we recommend the following optimizations:

- Move or install the database, Execution Server, and each agent on a separate machine.
- Do not install agents on the server. Agents perform task execution which increases the associated machine's CPU and memory usage.
- If you intend to use Automate at high capacity, switch to an Enterprise version of a database engine.

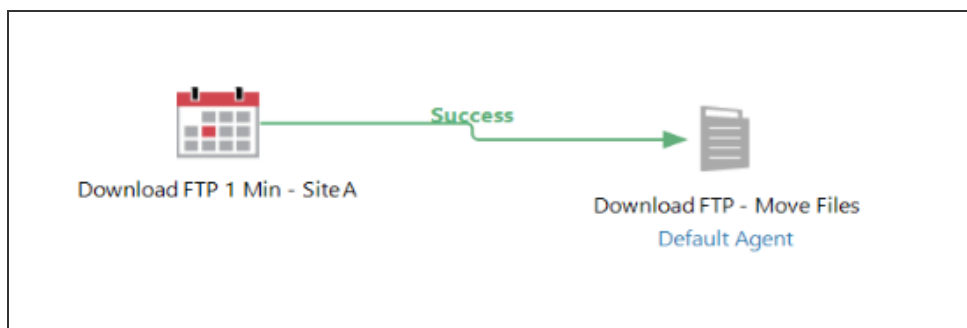
After applying these optimizations, the following diagram shows the improved design:



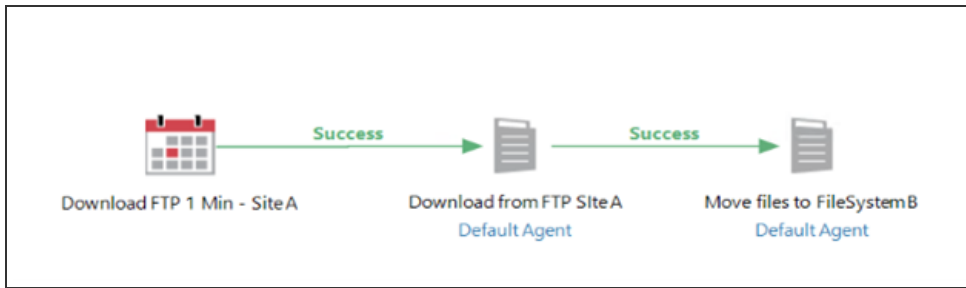
## How Automate runs scheduled jobs

To illustrate how Automate runs scheduled jobs, take an execution flow of a simple, repetitive job that runs every minute (for example, downloading files from an FTP server and then moving those files into a location).

Most customers follow the approach of using a workflow with a scheduled trigger that fires every minute:



Another approach is using two separate tasks to do each action separately; one to download the files, the other to move them (sometimes for reusability purposes):



Either approach starts with a trigger that fires by responding to its event, which results in creating an instance of the workflow. Once a workflow instance initiates (identified as transaction in the logs), the execution server orchestrates the flow of execution between each of the workflow's items (for example, tasks, evaluations, waits, etc.). If the flow leads to an executable (task or process), an instance of that executable gets created, wrapped by all its variables and settings, and then queued to run on an agent.

Once an agent picks a task to execute, it applies priorities to the task, and if all criteria is met, the task starts. The task will continue to run if the following conditions are met:

- The task did not complete
- Exceptions did not occur
- The task did not time out

When a task finishes execution, the status and results of that execution, along related data, are reported back to the Execution Server. Once the Execution Server receives the results, it releases all resources, and then continues with the next workflow item. This cycle will continue as long as:

- There are items to execute in the workflow
- Exceptions do not occur
- Workflow does not time out

The previous approaches are all valid, yet they hide heavy execution costs on the Execution Server and agent sides, which eventually results in excessive triggering. The more jobs added, the worst the symptoms become.

## Excessive triggering

In large scale environments, repetitive workflow triggering directly impacts Automate's performance and stability. Triggering is a costly operation, and many Automate developers need to consider the following factors:

- The number of triggers firing within a period
- The number of triggers using short intervals (for example, 1 second to 5 minutes)
- The number of workflows triggering by the same trigger

The performance threshold is also dependent on criteria, such as:

- Hardware specifications of the machine running the Execution Server (CPU, memory, etc.)
- Domain policies
- Network speed
- Database engine
- Third-party services

**IMPORTANT:** Third-party services are a key factor in determining a system's performance. For instance, the resources required to run a job that downloads files from an FTP service varies based on the FTP service vendor, frequency of the job, and the size of the files.

By keeping track of these factors, each environment can create its own threshold level.

## Why you should avoid excessive triggering

Once scheduling conditions are met, all workflows containing a trigger initiate. The more workflows you have triggering, the more system resources you will need to run those workflows.

For example, scheduling jobs that exceed what the system's resources can handle can result in various issues (for example, agent disconnect messages, etc.) This is an indication that the Execution Server is overloaded due to excessive triggering. Without the proper reporting and settings, issues like this can go undetected resulting in an unstable system.

A workaround to this issue is restarting the Automate services prior peak hours; however, you can avoid this by performing the following optimizations to the system:

- Utilization of the Execution Server resources to handle triggering, flow management, database transactions, etc.
- Reducing agent server communication
- Utilization of agent machine's resources
- Efficient logging

## How to detect and avoid excessive triggering

## Review trigger distribution periodically

Reduce the number of triggers firing within proximity of each other by changing when and how often those triggers fire. This promotes a less aggressive triggering schedule.

Run the [All Workflows Triggers Detail SQL query](#) periodically to identify patterns, such as:

- Substantial number of workflows triggering at once
- Triggers with short intervals (1 second to 5 minutes)
- Peaks and valleys in the triggering schedule

The following is a sample output of the All Workflows Triggers Detail SQL query which returns a list of all scheduled triggers within all available workflows:

| TriggerName           | Type    | FRQ | Last Launch Date | Next Launch Date | WorkflowName                                       | ConstructPath  | lastRun   |
|-----------------------|---------|-----|------------------|------------------|--|--|---|
| :18 Half Hour         | Minutes | 300 | 10/9/2022 14:18  | 10/9/2022 19:18  | Server Cleanup Routine                             | \\WORKFLOWS\\imaging\\Performance & Monitoring         | Workflow 'Server Cleanup Routine' completed successfully.                             |
| :18 Half Hour         | Minutes | 300 | 10/9/2022 14:18  | 10/9/2022 19:18  | FA-Image-Colo-FTP-Reprocess-Hourly 2019            | \\WORKFLOWS\\imaging\\Misc\\First American - REBOUND   | Workflow 'FA-Image-Colo-FTP-Reprocess-Hourly 2019' failed...                          |
| :28 Half Hour         | Minutes | 300 | 10/9/2022 14:28  | 10/9/2022 19:28  | TN-Wilson IMG (WEB ACQ)                            | \\WORKFLOWS\\imaging\\Go Forward\\Tennessee            | Workflow 'TN-Wilson IMG (WEB ACQ)' failed...  |
| :28 Half Hour         | Minutes | 300 | 10/9/2022 14:28  | 10/9/2022 19:28  | FL-Walton IMG                                      | \\WORKFLOWS\\imaging\\Go Forward\\Florida              | Workflow 'FL-Walton IMG' completed successfully.                                      |
| :40 Hour              | Minutes | 300 | 10/9/2022 14:33  | 10/9/2022 19:33  | MI-Process Batches                                 | \\WORKFLOWS\\Developers\\antony.yesudas\\MI Conversion | Workflow 'MI-Process Batches' failed...   |
| :40 Hour              | Minutes | 300 | 10/9/2022 14:33  | 10/9/2022 19:33  | FNFI FTP Transfer (non BK)                         | \\WORKFLOWS\\imaging\\Misc\\FNFI                       | Workflow 'FNFI FTP Transfer (non BK)' failed...                                       |
| QC Monitor (45 mts)   | Minutes | 45  | 10/9/2022 17:00  | 10/9/2022 17:45  | QC-Discrepancy Upload                              | \\WORKFLOWS\\imaging\\Misc\\Discrepancy                | Workflow 'QC-Discrepancy Upload' completed successfully.                              |
| QC Monitor (45 mts)   | Minutes | 45  | 10/9/2022 17:00  | 10/9/2022 17:45  | QC Monitor   | \\WORKFLOWS\\imaging\\Performance & Monitoring         | Workflow 'QC Monitor' completed successfully.   |
| OI Indexing Uploader  | Minutes | 30  | 10/9/2022 17:10  | 10/9/2022 17:40  | OI Indexing Uploader                               | \\WORKFLOWS\\imaging\\Misc\\OI Indexing                |   |
| FTP Download - LA - 1 | Minutes | 5   | 10/9/2022 17:20  | 10/9/2022 17:25  | FTP Download - LA - 1                              | \\WORKFLOWS\\imaging\\Misc\\Plant File Processor       | Workflow 'Plant File Processor' completed successfully.                               |
| FTP Download - LA - 2 | Minutes | 2   | 10/9/2022 17:20  | 10/9/2022 17:22  | FTP Download - LA - 2                              | \\WORKFLOWS\\imaging\\Misc\\DSM Extraction             | Workflow 'DSM Extraction (File Drop) - Cory' completed successfully.                  |
| FTP Download - LA - 3 | Minutes | 3   | 10/9/2022 17:20  | 10/9/2022 17:23  | FTP Download - LA - 3                              | \\WORKFLOWS\\imaging\\Misc\\DSM Extraction             | Workflow 'DSM Extraction (Milwaukee Deed Report Request)' failed...                   |
| FTP Download - LA - 4 | Minutes | 3   | 10/9/2022 17:20  | 10/9/2022 17:23  | FTP Download - LA - 4                              | \\WORKFLOWS\\imaging\\Misc\\DSM Extraction             | Workflow 'DSM Extraction (File Drop) - Tyson' failed...                               |
| every 4 hours         | Minutes | 200 | 10/9/2022 17:20  | 10/9/2022 20:40  | TX-Bexar IMG (New)                                 | \\WORKFLOWS\\imaging\\Go Forward\\Texas                |   |
| every 4 hours         | Minutes | 200 | 10/9/2022 17:20  | 10/9/2022 20:40  | TX-Starr IMG (FTP)                                 | \\WORKFLOWS\\imaging\\Go Forward\\Texas                | Workflow 'TX-Starr IMG (FTP)' completed successfully.                                 |
| :23 Half Hour         | Minutes | 30  | 10/9/2022 17:23  | 10/9/2022 17:53  | Rebound_MultiSource-Processor (AlphaDelegation) v2 | \\WORKFLOWS\\imaging\\Misc\\First American - REBOUND   | Workflow 'Rebound_MultiSource-Processor (AlphaDelegation) v2' completed successfully. |

In this sample, you can see that four workflows that were running repetitively within small intervals and they were identified by filtering and sorting the query results using the following:

- Sort by Last Launch Time - Sorting by this column groups together patterns of excessive triggering within a period.
- Filter by Minutes and Seconds, then sort by Frequency - Filtering by minutes and seconds shows triggers with small intervals (1 sec > 5 minutes). and then sorting the results by Frequency shows how often they ran.

For instance, a workflow set to trigger every three minutes has a high probability of overlapping, queuing, or failing. Without proper logging and monitoring, it is likely this could go unnoticed.

## Trigger reusability/bad distribution

During analysis we might find patterns indicating excessive use of trigger reusability or bad distribution.

For example, the following pattern appears in this sample report image:

- Daily at 1:00 AM, 26 workflows are scheduled to trigger at once
- The 26 workflows are then followed by 12 workflows to trigger daily at 1:05 AM

| Trigger | trigg | LastLaunchDate | NextLaunchDate  | WorkflowName                    | ConstructPath                           | LastRun  |
|---------|-------|----------------|-----------------|---------------------------------|---|--|
| 0:00:00 | Daily | 1/10/2022 0:00 | 10/10/2022 0:00 | DH-Allen (MG) (DAS)             | WORKFLOW\SinagGo\Fow and Ohio           | Workflow DH-Allen (MG) (DAS) completed successfully.             |
| 0:00:00 | Daily | 1/10/2022 0:00 | 10/10/2022 0:00 | DH-Tuscarawas (MG) (DAS)        | WORKFLOW\SinagGo\Fow and Ohio           | Workflow DH-Tuscarawas (MG) (DAS) completed successfully.        |
| 0:00:00 | Daily | 1/10/2022 0:00 | 10/10/2022 0:00 | DH-Pottawatomie (MG) (DAS)      | WORKFLOW\SinagGo\Fow and Oklahoma       | Workflow DH-Pottawatomie (MG) (DAS) completed successfully.      |
| 1:00:00 | Daily | 1/10/2022 1:00 | 10/10/2022 1:00 | FMT Volume Forecast             | WORKFLOW\SinagGo\Fow and FMT            | Workflow FMT Volume Forecast completed successfully.             |
| 1:00:00 | Daily | 1/10/2022 1:00 | 10/10/2022 1:00 | CO-Teller (MG) & GG (DAS)       | WORKFLOW\SinagGo\Fow and Colorado       | Workflow CO-Teller (MG) & GG (DAS) completed successfully.       |
| 1:00:00 | Daily | 1/10/2022 1:00 | 10/10/2022 1:00 | Maps - Daily Report             | WORKFLOW\SinagGo\Fow and Maps           | Workflow Maps - Daily Report completed successfully.             |
| 1:00:00 | Daily | 1/10/2022 1:00 | 10/10/2022 1:00 | NV-Guest (MG)                   | WORKFLOW\SinagGo\Fow and West Virginia  | Workflow NV-Guest (MG) completed successfully.                   |
| 1:00:00 | Daily | 1/10/2022 1:00 | 10/10/2022 1:00 | AZ-Coconino AOV                 | WORKFLOW\SinagGo\Fow and Arizona        | Embedded Workflow AZ-Coconino AOV completed successfully.        |
| 1:00:00 | Daily | 1/10/2022 1:00 | 10/10/2022 1:00 | NC-Catawba (MG)                 | WORKFLOW\SinagGo\Fow and North Carolina | Workflow NC-Catawba (MG) completed successfully.                 |
| 1:00:00 | Daily | 1/10/2022 1:00 | 10/10/2022 1:00 | DH-Manroe (MG)                  | WORKFLOW\SinagGo\Fow and Ohio           | Workflow DH-Manroe (MG) completed successfully.                  |
| 1:00:00 | Daily | 1/10/2022 1:00 | 10/10/2022 1:00 | SC-Newberry (MG)                | WORKFLOW\SinagGo\Fow and South Carolina | Workflow SC-Newberry (MG) completed successfully.                |
| 1:00:00 | Daily | 1/10/2022 1:00 | 10/10/2022 1:00 | VA-Prince William (MG) (FTP)    | WORKFLOW\SinagGo\Fow and Virginia       | Workflow VA-Prince William (MG) (FTP) completed successfully.    |
| 1:00:00 | Daily | 1/10/2022 1:00 | 10/10/2022 1:00 | NV-Clark (MG)                   | WORKFLOW\SinagGo\Fow and Nevada         | Workflow NV-Clark (MG) completed successfully.                   |
| 1:00:00 | Daily | 1/10/2022 1:00 | 10/10/2022 1:00 | DH-Lake (MG) (DAS)              | WORKFLOW\SinagGo\Fow and Ohio           | Workflow TX-Williamson (MG) completed successfully.              |
| 1:00:00 | Daily | 1/10/2022 1:00 | 10/10/2022 1:00 | DH-Bloom (MG)                   | WORKFLOW\SinagGo\Fow and Ohio           | Workflow TX-Williamson (MG) completed successfully.              |
| 1:00:00 | Daily | 1/10/2022 1:00 | 10/10/2022 1:00 | DH-Fulton (MG)                  | WORKFLOW\SinagGo\Fow and Ohio           | Workflow TX-Williamson (MG) completed successfully.              |
| 1:00:00 | Daily | 1/10/2022 1:00 | 10/10/2022 1:00 | DH-Coshocoon (MG)               | WORKFLOW\SinagGo\Fow and Ohio           | Workflow TX-Williamson (MG) completed successfully.              |
| 1:00:00 | Daily | 1/10/2022 1:00 | 10/10/2022 1:00 | NH-Curry (MG) (DAS)             | WORKFLOW\SinagGo\Fow and Texas          | Workflow TX-Williamson (MG) completed successfully.              |
| 1:00:00 | Daily | 1/10/2022 1:00 | 10/10/2022 1:00 | KY-Cimenden (MG) (COLO-FTP)     | WORKFLOW\SinagGo\Fow and Kentucky       | Workflow KY-Cimenden (MG) (COLO-FTP) completed successfully.     |
| 1:00:00 | Daily | 1/10/2022 1:00 | 10/10/2022 1:00 | VA-San Juan (MG) (File T)       | WORKFLOW\SinagGo\Fow and Washington     | Workflow VA-San Juan (MG) (File T) failed.                       |
| 1:00:00 | Daily | 1/10/2022 1:00 | 10/10/2022 1:00 | OR-Mason (MG)                   | WORKFLOW\SinagGo\Fow and Oregon         | Workflow OR-Mason (MG) failed.                                   |
| 1:00:00 | Daily | 1/10/2022 1:00 | 10/10/2022 1:00 | KS-Leavenworth (MG) (FTP)       | WORKFLOW\SinagGo\Fow and Kansas         | Workflow KS-Leavenworth (MG) (FTP) completed successfully.       |
| 1:00:00 | Daily | 1/10/2022 1:00 | 10/10/2022 1:00 | WY-Sublette (MG) (DAS)          | WORKFLOW\SinagGo\Fow and Wyoming        | Workflow WY-Sublette (MG) (DAS) failed.                          |
| 1:00:00 | Daily | 1/10/2022 1:00 | 10/10/2022 1:00 | WV-Chapman (MG) (FTP)           | WORKFLOW\SinagGo\Fow and West Virginia  | Workflow WV-Chapman (MG) (FTP) completed successfully.           |
| 1:00:00 | Daily | 1/10/2022 1:00 | 10/10/2022 1:00 | COLO to EP Cache Trans          | WORKFLOW\SinagGo\Fow and Colorado       | Workflow COLO to EP Cache Trans failed.                          |
| 1:00:00 | Daily | 1/10/2022 1:00 | 10/10/2022 1:00 | CA-Modoc (MG) (FTP)             | WORKFLOW\SinagGo\Fow and California     | Workflow CA-Modoc (MG) (FTP) completed successfully.             |
| 1:00:00 | Daily | 1/10/2022 1:00 | 10/10/2022 1:00 | FL-Polk (MG) (FTP) (new)        | WORKFLOW\SinagGo\Fow and Florida        | Workflow FL-Polk (MG) (FTP) (new) completed successfully.        |
| 1:00:00 | Daily | 1/10/2022 1:00 | 10/10/2022 1:00 | TX-Mengrove (MG) & GG           | WORKFLOW\SinagGo\Fow and Texas          | Workflow TX-Mengrove (MG) & GG completed successfully.           |
| 1:00:00 | Daily | 1/10/2022 1:00 | 10/10/2022 1:00 | MT-Powder River (MG) (COLO-FTP) | WORKFLOW\SinagGo\Fow and Montana        | Workflow MT-Powder River (MG) (COLO-FTP) completed successfully. |
| 1:05:00 | Daily | 1/10/2022 1:05 | 10/10/2022 1:05 | IL-Pike (MG) (COLO-FTP)         | WORKFLOW\SinagGo\Fow and Illinois       | Workflow IL-Pike (MG) (COLO-FTP) failed.                         |
| 1:05:00 | Daily | 1/10/2022 1:05 | 10/10/2022 1:05 | CA-Santa Clara (MG)             | WORKFLOW\SinagGo\Fow and California     | Workflow CA-Santa Clara (MG) failed.                             |
| 1:05:00 | Daily | 1/10/2022 1:05 | 10/10/2022 1:05 | NC-Durham (MG)                  | WORKFLOW\SinagGo\Fow and North Carolina | Workflow NC-Durham (MG) completed successfully.                  |
| 1:05:00 | Daily | 1/10/2022 1:05 | 10/10/2022 1:05 | FL-Hillsborough (MG) (FTP)      | WORKFLOW\SinagGo\Fow and Florida        | Workflow FL-Hillsborough (MG) (FTP) completed successfully.      |
| 1:05:00 | Daily | 1/10/2022 1:05 | 10/10/2022 1:05 | MT-Yellowstone (MG) (COLO-FTP)  | WORKFLOW\SinagGo\Fow and Montana        | Workflow MT-Yellowstone (MG) (COLO-FTP) completed successfully.  |
| 1:05:00 | Daily | 1/10/2022 1:05 | 10/10/2022 1:05 | CA-Sierra GG                    | WORKFLOW\SinagGo\Fow and California     | Workflow CA-Sierra GG completed successfully.                    |
| 1:05:00 | Daily | 1/10/2022 1:05 | 10/10/2022 1:05 | NC-Richmond (MG)                | WORKFLOW\SinagGo\Fow and North Carolina | Workflow NC-Richmond (MG) completed successfully.                |
| 1:05:00 | Daily | 1/10/2022 1:05 | 10/10/2022 1:05 | MT-Owl Lodge (MG) (COLO-FTP)    | WORKFLOW\SinagGo\Fow and Montana        | Workflow MT-Owl Lodge (MG) (COLO-FTP) failed.                    |
| 1:05:00 | Daily | 1/10/2022 1:05 | 10/10/2022 1:05 | FL-Charlotte (MG) (Colo-FTP)    | WORKFLOW\SinagGo\Fow and Florida        | Workflow FL-Charlotte (MG) (Colo-FTP) failed.                    |
| 1:05:00 | Daily | 1/10/2022 1:05 | 10/10/2022 1:05 | DH-Hocking (MG) (DAS)           | WORKFLOW\SinagGo\Fow and Ohio           | Workflow DH-Hocking (MG) (DAS) failed.                           |
| 1:05:00 | Daily | 1/10/2022 1:05 | 10/10/2022 1:05 | DH-Henry (MG)                   | WORKFLOW\SinagGo\Fow and Ohio           | Workflow DH-Henry (MG) failed.                                   |
| 1:05:00 | Daily | 1/10/2022 1:05 | 10/10/2022 1:05 | NV-Monroe (MG) (DAS)            | WORKFLOW\SinagGo\Fow and New York       | Workflow NV-Monroe (MG) (DAS) failed.                            |
| 1:45:00 | Daily | 1/10/2022 1:45 | 10/10/2022 1:45 | DH-Mahoning (MG) (DAS)          | WORKFLOW\SinagGo\Fow and Ohio           | Workflow DH-Mahoning (MG) (DAS) completed successfully.          |
| 1:45:00 | Daily | 1/10/2022 1:45 | 10/10/2022 1:45 | NV-Clark (MG)                   | WORKFLOW\SinagGo\Fow and Nevada         | Workflow NV-Clark (MG) completed successfully.                   |
| 1:45:00 | Daily | 1/10/2022 1:45 | 10/10/2022 1:45 | DH-Lake (MG) (DAS)              | WORKFLOW\SinagGo\Fow and Ohio           | Workflow DH-Lake (MG) (DAS) failed.                              |
| 1:45:00 | Daily | 1/10/2022 1:45 | 10/10/2022 1:45 | DH-Bloom (MG)                   | WORKFLOW\SinagGo\Fow and Ohio           | Workflow DH-Bloom (MG) completed successfully.                   |
| 1:45:00 | Daily | 1/10/2022 1:45 | 10/10/2022 1:45 | DH-Fulton (MG)                  | WORKFLOW\SinagGo\Fow and Ohio           | Workflow DH-Fulton (MG) failed.                                  |
| 1:45:00 | Daily | 1/10/2022 1:45 | 10/10/2022 1:45 | DH-Coshocoon (MG)               | WORKFLOW\SinagGo\Fow and Ohio           | Workflow DH-Coshocoon (MG) failed.                               |
| 1:45:00 | Daily | 1/10/2022 1:45 | 10/10/2022 1:45 | NH-Curry (MG) (DAS)             | WORKFLOW\SinagGo\Fow and New Mexico     | Workflow NH-Curry (MG) (DAS) failed.                             |
| 1:45:00 | Daily | 1/10/2022 1:45 | 10/10/2022 1:45 | DH-Adams (MG)                   | WORKFLOW\SinagGo\Fow and Ohio           | Workflow DH-Adams (MG) failed.                                   |

Checking other intervals such as minutes, hours, etc., can reveal a large number of repetitive triggers within the same period, indicating a peak around 1:00 AM.

In another example, the pattern below shows a large number of triggers with small intervals. It is a best practice to review workflow designs and trigger intervals, and optimize them using the approaches below:



| TriggerName              | TriggerType | Frequency | LastLaunchDate | NextLaunchDate | WorkflowName                     |
|--------------------------|-------------|-----------|----------------|----------------|----------------------------------|
| 39 :18 Half Hour         | Minutes     | 60        | 10/9/20        | 1:00 10/9/20   | 2:00 Server Cleanup Routine      |
| 41 :18 Half Hour         | Minutes     | 60        | 10/9/20        | 1:00 10/9/20   | 19:18 FA-Image-Colo-FTP-Reproce  |
| 70 :28 Half Hour         | Minutes     | 60        | 10/9/20        | 1:00 10/9/20   | 19:28 TN-Wilson IMG (WEB ACQ)    |
| 71 :28 Half Hour         | Minutes     | 10        | 10/9/20        | 1:00 10/9/20   | 19:28 FL-Walton IMG              |
| 90 :40 Hour              | Minutes     | 7         | 10/9/20        | 1:00 10/9/20   | 19:33 MI-Process Batches         |
| 99 :40 Hour              | Minutes     | 5         | 10/9/20        | 1:00 10/9/20   | 19:33 FNF1 FTP Transfer (non BK) |
| 17 QC Monitor (45 mts)   | Minutes     | 5         | 10/9/20        | 1:00 10/9/20   | 17:45 QC-Discrepancy Upload      |
| 20 QC Monitor (45 mts)   | Minutes     | 5         | 10/9/20        | 1:00 10/9/20   | 17:45 QC Monitor                 |
| 74 OI Indexing Uploader  | Minutes     | 2         | 10/9/20        | 1:01 10/9/20   | 17:40 OI Indexing Uploader       |
| 11 FTP Download - LA - 1 | Minutes     | 5         | 10/9/20        | 1:01 10/9/20   | 17:25 FTP Download - LA - 1      |
| 12 FTP Download - LA - 2 | Minutes     | 2         | 10/9/20        | 1:01 10/9/20   | 17:22 FTP Download - LA - 2      |
| 13 FTP Download - LA - 3 | Minutes     | 3         | 10/9/20        | 1:02 10/9/20   | 17:23 FTP Download - LA - 3      |
| 14 FTP Download - LA - 4 | Minutes     | 3         | 10/9/20        | 1:02 10/9/20   | 17:23 FTP Download - LA - 4      |
| 16 every 4 hours         | Minutes     | 3         | 10/9/20        | 1:02 10/9/20   | 10:40 TX-Bexar IMG (New)         |
| 17 every 4 hours         | Minutes     | 3         | 10/9/20        | 1:02 10/9/20   | 10:40 TX-Starr IMG (FTP)         |
| 19 :23 Half Hour         | Minutes     | 3         | 10/9/20        | 1:02 10/9/20   | 17:53 Rebound_MultiSource-Proci  |

The reports above show a high probability of system instability, resulting in the following behavior:

- A large number of jobs queuing on the Execution Server machine, waiting to be picked up by agents
- Jobs getting stuck on an agent machine due to a lack of available system resources
- The Execution Server becoming overloaded by growing numbers of new requests
- New jobs queuing while triggered jobs remain unfinished

The previous state of the system eventually causes communication failures between the Execution Server and agent(s). By design, the agent tries to reestablish a new connection with the Execution Server which can leave previously unfinished jobs in an incomplete state.

It is inevitable that the Execution Server and agent will become overloaded by those patterns, as the number of new triggers will exceed the number of workflows finishing execution. The key is to distribute the balancing load effectively. A best practice is to use the following formula for determining the interval of any scheduled trigger:

$$\text{Interval} > \text{average workflow execution time} + 2 \text{ minutes}$$

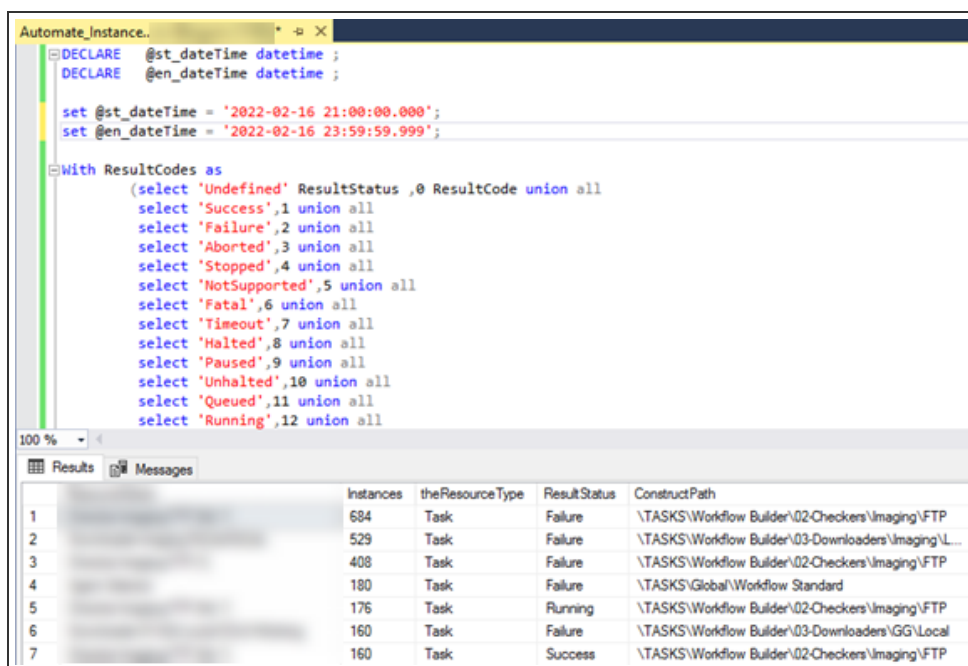
To find the right intervals, manually run a job during peak hours, and then use its execution duration value (this applies to development, staging, and production environments). This will yield an ideal initial value for the average workflow execution time for you to review later by analyzing the execution logs.

## Monitor the number of instances created

One of the main causes for system instability is when triggers with small intervals create an excessive number of simultaneous task instances. Instance counts lead into detecting those types of behaviors and you need to adjust them by changing the workflow design.

To see the total number of instances created within a period of time on your system, run the [Instances Count Basic SQL query](#). High numbers appearing in the query results indicate excessive triggering for a workflow or task. Using this query is an effective way to identify workflows and tasks that trigger frequently, and then start optimizing them (if applicable). We recommend starting with the workflows containing the smallest intervals. For more information, see [Optimum workflow designs on page 22](#). This query will also show the following:

- The count, the type of resource, the state of the execution (that is, failure, success, or running), and the location of the workflow and task for easy access. It is essential to keep the numbers reasonable, depending on your organization's needs.
- If there is gap between the total number of instances reported as Running vs. Failure vs. Successful. These numbers should be close, but major differences mean workflows and tasks are stalling or failing frequently.



```

DECLARE @st_datetime datetime;
DECLARE @en_datetime datetime;

set @st_datetime = '2022-02-16 21:00:00.000';
set @en_datetime = '2022-02-16 23:59:59.999';

With ResultCodes as
(
select 'Undefined',0 ResultCode union all
select 'Success',1 union all
select 'Failure',2 union all
select 'Aborted',3 union all
select 'Stopped',4 union all
select 'NotSupported',5 union all
select 'Fatal',6 union all
select 'Timeout',7 union all
select 'Halted',8 union all
select 'Paused',9 union all
select 'Unhalted',10 union all
select 'Queued',11 union all
select 'Running',12 union all
)

```

|   | Instances | theResourceType | ResultStatus | ConstructPath  |
|---|-----------|-----------------|--------------|--|
| 1 | 684       | Task            | Failure      | \\TASKS\\Workflow Builder\\02-Checkers\\Imaging\\FTP     |
| 2 | 529       | Task            | Failure      | \\TASKS\\Workflow Builder\\03-Downloaders\\Imaging\\L... |
| 3 | 408       | Task            | Failure      | \\TASKS\\Workflow Builder\\02-Checkers\\Imaging\\FTP     |
| 4 | 180       | Task            | Failure      | \\TASKS\\Global\\Workflow Standard                       |
| 5 | 176       | Task            | Running      | \\TASKS\\Workflow Builder\\02-Checkers\\Imaging\\FTP     |
| 6 | 160       | Task            | Failure      | \\TASKS\\Workflow Builder\\03-Downloaders\\GG\\Local     |
| 7 | 160       | Task            | Success      | \\TASKS\\Workflow Builder\\02-Checkers\\Imaging\\FTP     |

## Resource locks

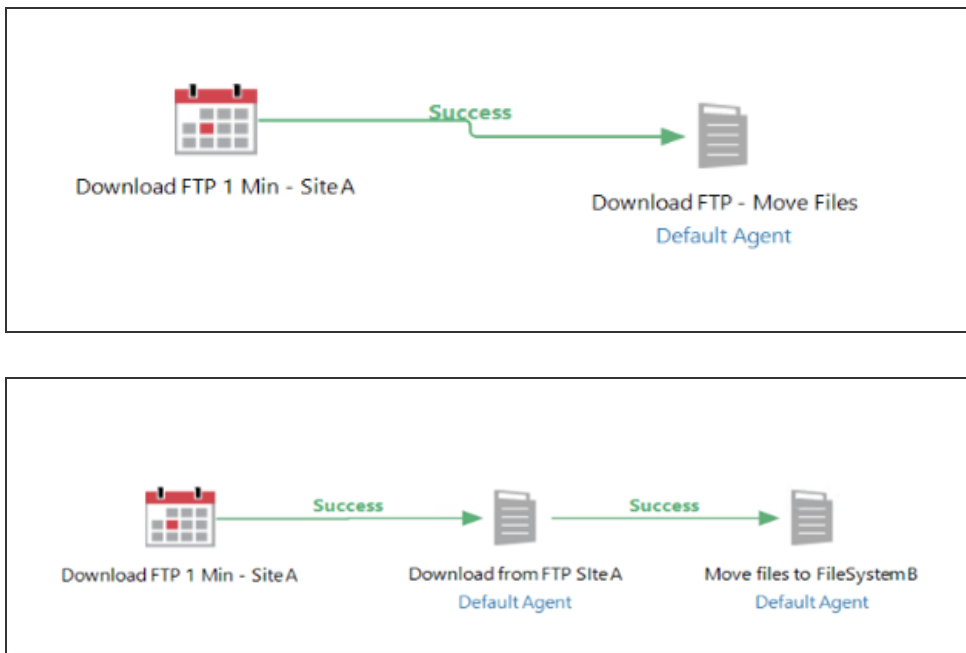
Simultaneously executing tasks that access the same resources (for example, file access, FTP, email, etc.) can result in locking those resources, which can then result in tasks stalling.

## Optimum workflow designs

In this section, the goal is to introduce more efficient workflow design patterns to current and new jobs.

## Workflow loops

In [How Automate runs scheduled jobs on page 16](#), we started from a basic workflow design for the job described, also known as a simple loop in the following images:



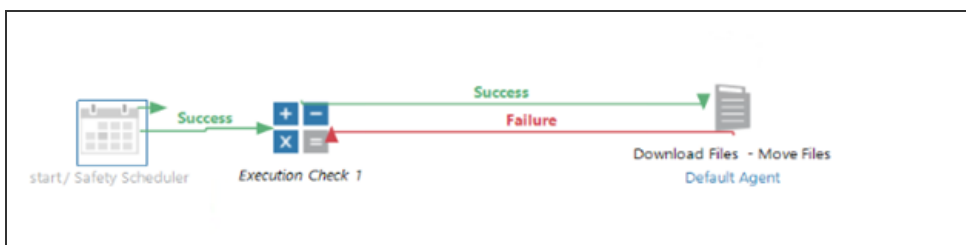
Both approaches are valid to address the job's need, yet this simple loop is one of the most expensive approaches with running jobs, especially when:

- The workflow runs frequently
- Tasks within a workflow might take longer than the interval of the trigger
- The job is divided into multiple small tasks (for example, reusability, the need to execute parts of the job on different agents, or agent groups)
- The priorities/timeouts are not set correctly

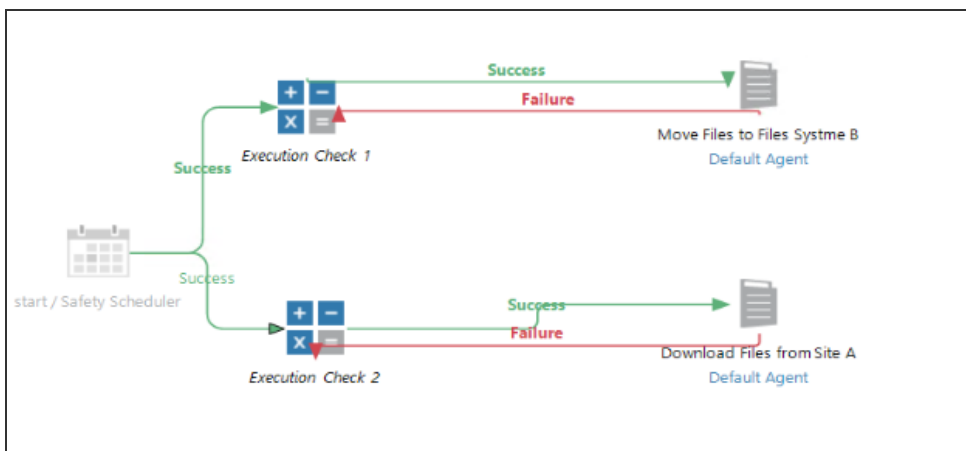
If you intend a workflow to perform a job repetitively, use loops within tasks to avoid the workflow triggering frequently which reduced the number of instances created.

The following is an alternative approach that utilizes an internal loop within the task. Applying this approach to the workflow loop design can look like the following examples:

1. Consolidating the logic into one task/agent. Download [Optimized Workflow - Single Task](#) to view this workflow on your system.



2. Executing the logic on two different tasks/agents. Download [Optimized Workflow - Two Tasks](#) to view this workflow on your system.



In these templates, the logic executes in a task infinitely until a condition is met or the task fails. The workflow automatically re-initiates the tasks in this design if the task fails/stops with respect to its priority setting. The Safety Scheduler trigger ensures there is a version of this workflow running, and the task's priority setting ensures there is only one instance of that task on the agent. The Safety Scheduler trigger's interval determines how often the execution status is checked. These changes step back the aggressiveness of the trigger condition, setting a more manageable interval.

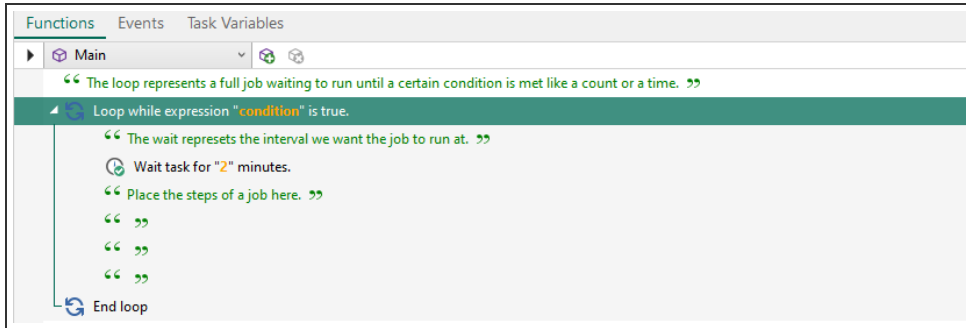
Since the goal is to reduce excessive schedule triggering in this example, we recommend setting the Safety Scheduler trigger to fire in minutes (10+).

Refrain from using workflow loops to check the connection status of an agent. Using workflow loops with small intervals adds significant overhead. To monitor the status of an agent or another resource, use the optimized task loop approach instead.

## Task loops

In [Workflow loops on page 23](#), we finished setting up the workflow. The next step in the optimization process is to take the steps from the task(s) shown in [How Automate runs scheduled jobs on page 16](#) and put it in the internal loop of this task design (see image). This loop repeats the job on the agent side until either a condition is met, or the job fails.

The internal loop's wait time or conditions should reflect the corresponding scheduled interval of the workflow being optimized.



The template will handle its own task failures and restart a new instance automatically. This requires exception handling logic to follow up with the task's failures, either at the task level, step level, or workflow level. It is best practice to add error notifications at the workflow and task level.

## Additional optimization settings

The following parameters and report can help with monitoring the behaviors discussed so far.

### Workflow and Workflow Item Timeout

The **Use Timeout** values should be greater than the maximum job execution time and intervals used.

### Reports > Workflow Durations

To help with preventing workflow overlap executions and better determine schedule intervals to account for a full cycle of executions, we recommend Automate developers review the Min and Max times in the **Workflow Durations** report.

### Priorities

Priorities control simultaneous tasks running on an agent.

### Task Priority

Specifies whether the task must run exclusively of other tasks and how to handle situations where another task is already running.

☐ Always run task  
☒ Run task if the number of running instances of this task is below the threshold

Running instances threshold:

1

If the condition above is not met:

☐ Hold task until condition is met  
☐ Hold task, then abort task if still not met  
☐ Hold task, then interrupt all running tasks and run task  
☐ Interrupt all running tasks and run task  
☐ Interrupt all running instances of this task and run task  
☒ Do not run task

If the task does not run, treat as:

☐ Success  
☒ Failure

☐ Run task if the number of running instances of all tasks is below the threshold  
☐ Run task if no other tasks are running on the agent

When initially designing a system or optimizing it, we recommend setting the **Running instances threshold** parameter to **1**, and then increase as needed.

Setting the **If the task does not run, treat as** parameter to **Failure** when the number of instances exceed a certain threshold during the optimization work, helps the Automate developer audit triggering cycles in the execution logs.

**IMPORTANT:** Running simultaneous task instances increases the probability of resource locks, which can lead to tasks stalling at the agent side, delayed start for scheduled jobs, and potentially creating a bottleneck at the Execution Server or agent machine. Once the system reaches a stable state, users can increase the threshold as needed to meet their organization's needs. You may need to perform additional optimizations if the new threshold creates the same bottleneck scenario.

## Agent Group Distribution

When using agent groups, set the appropriate Agent Group Distribution option. This parameter can have the most impact on the load directed to specific agents and each option has pros and cons.

For example, if you want a successful execution of tasks, then select **Run on each agent in order until a successful result** but ensure the order of the agents within the group is set correctly.

## Agent Group Distribution

Information regarding agent group subsets here.

- ☐ Run on all agents
- ☒ Run on each agent in order until a successful result
- ☐ Run on a single agent in round-robin fashion
- ☐ Run on a single agent in round-robin fashion, ignoring offline agents
- ☐ Run on the agent with the most free CPU capacity
- ☐ Run on all agents that meet the performance metrics

In another example, if the goal is to have a balanced load, then **Run on a single agent in round-robin fashion** will be more suitable.

Selecting **Run on all agents** can lead to potential redundancy in task execution for all agents, creating more instances which leads to overloaded system.

Consider all options based on the following:

- The goal of having this logic in place (success vs balancing)
- The system resources of the agent machine
- Dedicated agents for certain tasks only