FORTRA

Advanced Workflow Module for EFT v8.2 User Guide

202404010400

Table of Contents

Advanced Workflow Module

With EFT Event Rules, you can configure EFT to perform an Action automatically when a specific Event occurs. You can use Advanced Workflows to design scripts, batch files, macros, or any other code-intensive process using an easy drag-and-drop interface, and than add the Workflow to one or more Event Rules. Advanced Workflow Task names can appear in reports and logs.

The topics below describe using the Advanced Workflow actions specifically in EFT Event Rules.

> **NOTE:** For details of the Advanced Workflow actions, please refer to the help documentation in the Task Builder in the EFT administration interface. The help documentation for Advanced Workflow Actions can also be found in the installation folder for AutoMate: **C:\Program Files\Globalscape\AutoMate\Help\Default.htm**.

What's New in the AWM?

The Advanced Workflow Module uses components from Fortra's Automate Desktop, which is bundled in the EFT installer. This topic discusses changes from Automate 10 (the previous version bundled in the EFT installer) and Automate 2024.

> **IMPORTANT:** Before installing Automate 24.x or Automate Desktop 24.x, confirm your version of Windows is compatible with .NET Framework 4.8.

New Features

- 64-bit installer (As of version 11.7.1, Automate is only available as a 64-bit installer.)
- Dataset
    - Adds the ability to create and manipulate datasets, as well as manage the rows and columns within them.
- JSON Object

- JSON Decode - convert a JSON-encoded string into Automate variable engine objects.
- JSON Encode - convert an Automate variable engine object into a JSON-encoded string.

Enhancements

- Added support for Windows Server 2022
- Azure Action Enhancements
  - The Azure Action has been updated to support TLS 1.2.
- SharePoint Action Enhancements
  - SharePoint activities can now pull over information from multi-choice Lookup, User, and Group columns to populate datasets.
  - Added Azure AD Delegated and Interactive authentication types to support multifactor authentication (MFA) and single sign-on (SSO) with SharePoint Online-based connections.
  - Added SharePoint App Only and Azure AD App Only authentication types to support modern authentication with SharePoint Online.
  - Removed support for SharePoint 2007 and 2010 from the SharePoint action and SharePoint Browser.
  - Added the "Preserve internal column names" parameter to the SharePoint (Get files, Get Folders, and Get list item(s)) activities which provides the option to prevent SharePoint columns from being renamed in datasets.
  - SharePoint activities now correctly fail if user credentials are not valid.
  - The SharePoint (Get folders) activity no longer displays the "Error VirusStatus column already exists" error message during runtime for SharePoint 2019 and Online-based connections.
  - The SharePoint (Upload file(s)) activity now provides the option to set metatags while uploading files and retrieve uploaded file IDs.

  - The SharePoint Condition no longer displays a "not supported" error when set to "Evaluate condition immediately."
  - The SharePoint Trigger now properly fires tasks and workflows.

    > **NOTE:** The SharePoint trigger has been updated to use Transmission Control Protocol (TCP). The default TCP port is 9705. If you need to change this port number, please contact Fortra Support.

- Variable Action Enhancements
  - Added the ability to add a variable to the Watchlist by right-clicking on it from the Variables pane in the Task Builder.
  - Numeric-type variables are now handled and stored in decimals to provide higher precision.
  - The Variable (Create) and Variable (Set) activities now dynamically resize the "Initial value" and "New value" fields, based on input data.
  - The Variable (Create) activity now provides the option to secure a variable which will encrypt the value of a variable while editing a task.
  - Secured variables value will still be viewable in plaintext during runtime execution.
  - Constant Values can now be assigned while creating a Task Variable.
  - The Variable Engine has been optimized to improve overall system performance.

    > **IMPORTANT:** IMPORTANT: The optimizations to the Variable Engine now require all expression functions to contain () (for example, %Now()%).

- Welcome tab
  - Added new Welcome tab to the Task Administrator in Automate Desktop and the Server Management Console in Automate Plus/Ultimate to provide quick access to important Automate community aspects — all in one place.
  - In the event the Task Administrator in Automate Desktop or the Server Management Console in Automate Plus/Ultimate become disconnected, the

Welcome page will display a Connect button that allows the connection to be re-established.

- User Interface Enhancements
  - Rebranded the Automate and Automate Desktop user interfaces and company logos from HelpSystems to Fortra.
  - The user interface for Automate has been updated to provide an improved user experience.
  - Automate Desktop now displays the Automate Desktop product name in the user interface.
  - Task Builder now allows the option to automatically save tasks with an associated file every two minutes.
  - Added new UI Automation accessibility engine to support greater discoverability and ease of use when interacting with controls in desktop applications, such as Microsoft Excel.
  - Data value highlighting
  - Dockable panels
  - Variable inspector
  - Simplified event and function editing
- FTP Action Enhancements
  - The FTP (Logon) activity now supports an SSL/TLS (implicit) connection with TLS 1.0 disabled and TLS 1.1 and 1.2 enabled.
  - FTPS connections now work with TLS 1.2.
  - FTP - Logon activity now supports ECDH and ECDSA algorithms/ciphers.
  - FTP - Logon activity now supports AES (CTR,CBC) algorithms/ciphers.
  - The FTP (Upload file(s)) activity now provides the option to use multi-threaded uploads.
  - Added the ability to adjust SFTP pipeline length, block transfers, and upload/download block size.

    - Added the ability to manually adjust the "Upload buffer size (bytes)" setting for SFTP in the FTP (Logon, Download file(s), and Upload file(s)) activities.
    - Added the ability to suppress additional operations for SFTP transfers.
    - The FTP action now supports TLS 1.3 and the following TLS 1.3 ciphers:
        - TLS_AES_128_GCM_SHA256
        - TLS_AES_256_GCM_SHA384
        - TLS_CHACHA20_POLY1305_SHA256
    - The FTP action now supports the following SFTP ciphers:
        - EA_3DES was renamed to EA_3DES-CBC
        - EA_AES128 was renamed to EA_AES128_CBC
        - EA_AES192 was renamed to EA_AES192_CBC
        - EA_AES256 was renamed to EA_AES256_CBC
- OCR Action Enhancements
    - Added support for the Turkish language for all OCR activities.
    - Added support for the Tesseract engine to improve accuracy and provide more reliable results.
    - Added JPEG image format support to all OCR activities.
    - Added support for the Hebrew language to all OCR activities.
    - Added support for the Japanese language to all OCR activities.
- File System Action Enhancements
    - The File System (Move) activity's Source and Destination parameters now support over 260 characters.
    - Added the ability to encode/decode a file using Base64 encoding by way of the new File System (Base64 encode/decode) activity.
    - The File System (Get information) activity now provides the ability to include checksums for all files.

  - The File System (Dataset to CSV) activity now supports enclosing dataset cell values in double quotes (") in CSV files.
  - Improved the performance of the File System (Dataset to CSV) activity.
- Terminal Action Enhancements
  - Added the ability to select the terminal window resolution for TN3270 and TN5250 emulations to the Terminal (Connect) activity.
  - Added the ability to attempt to automatically signoff from a terminal before disconnecting for TN3270 and TN5250 emulations to the Terminal (Connect) activity.
  - Added the ability to change font sizes for TN3270 and TN5250 emulations to the Terminal (Connect) activity.
  - Terminal - Connect activity supports public key authentication method.
- Image Action Enhancements
  - The Image action now supports .tif and .jpeg file formats.
- Compression Action Enhancements
  - Added the ability to verify if compressed files are valid and not corrupted by way of the new Compression (Verify) activity.
- HTTP Action Enhancements
  - Added the ability to modify existing data or a file on the specified server by way of the new HTTP (Patch) activity.
  - The HTTP (Post) activity now provides the ability to submit HTTP Post requests that do not contain a body.
  - The HTTP (Get) activity now accepts the "Create and Populate Dataset" field as a valid action output without an associated variable or output file.
  - Added the ability to create predefined HTTP connections using the new HTTP (Define) activity. Users can define HTTP connection and authentication information in a single place and then reuse it through a task in various HTTP steps.

  - Added support for additional authentication means (OATH 2.0, Bearer Token, API Key).
  - Corrected numerous issues involving JSON decoding.
- Exchange Action Enhancements
  - Added support for Exchange Online with Modern Authorization to the Email Server Settings.
  - Added support for Exchange Online with Modern Authorization to the Email Default Task Agent Properties.
  - Added support for Exchange Online with Modern Authorization to the Email Default Properties.
  - The hot key for each Exchange activity is now unique.
  - The Exchange (Get object(s)) activity now returns an empty dataset and no longer fails if no objects exist.
  - The Exchange Action as well as all other Actions and Automate/Enterprise components that employ exchange functionality have been updated to support TLS 1.2.
- PDF Action Enhancements
  - Added the ability for the PDF (Insert) activity to perform an insert at the last page of a PDF document.
  - Added new annotation management activities (Add annotation, Delete annotation (s), and List annotation(s)).
  - Added the ability to insert an image as a new page in a PDF in the PDF (Insert) activity.
  - Corrected numerous issues with the PDF action.
  - The PDF (Get attachment(s)) activity now provides the option to overwrite existing file attachments with the same name or individually save them with unique file names.

    - The PDF (Extract) activity now provides the option to extract contents into a single, multi-page TIFF file.
    - The PDF (Create) activity now provides the option to change page orientation.
- Email Action Enhancements
    - The Email (Send Message) activity can now create a dataset to capture information regarding each email sent.
- Dialog Action Enhancements
    - The Dialog (Open File) activity now provides improved file and folder browsing navigation.
- BASIC Script Action Enhancements
    - The BASIC Script (Execute) activity's Embedded text box has been increased in size.
- Web Service Action Enhancements
    - Web Service action has been updated to support TLS 1.2.

Installing the Advanced Workflow Module

The Advanced Workflow module (AWM) is installed with Fortra's EFT Server and is included in the 15-day trial of EFT, during which you can create, edit, and execute powerful workflows using the EFT Event Rule system. After the trial has expired, you can still create new workflows and edit existing ones; however, after the trial expires, you will no longer be able to execute the workflows with EFT Event Rules. After you purchase and activate a serial number for the module, you will once again be able to use the workflows in Event Rules. The workflows are not deleted until you delete them.

EFT subscription licenses have a week-long grace period in which the module will continue to function after being expired. Automate does not have this week-long grace period. When the Advanced Workflow module (AWM) expires from a subscription license and enters its grace period, the Automate side will expire with no grace period; the AWM module will stop functioning in its grace period.

IMPORTANT: If you are upgrading from a version of EFT before v8.2, please read Upgrading the Advanced Workflow Module. There are numerous items that you need to know BEFORE you upgrade, such as the conversion of Advanced Workflow AML files.

Automate Desktop Service

The Automate Desktop service enables tasks created using Automate Desktop to trigger automatically on the computer. If this service is stopped, Automate Desktop tasks will not be run at their scheduled times.

Ensure that the Automate Service Account is enabled on the **Server > Administration** tab in the EFT administration interface to continue to run event rules when the Windows session is logged out or locked or EFT server service account is logged out.

> **NOTE:** If no Windows user account is provided and enabled for the Automate service account, EFT Event Rules with Advanced Workflows will not properly execute when the Windows session is locked or logged off. (The EFT event rule will trigger, however, the Automate task will fail.)

Before creating or using Advanced Workflows in EFT, you should wait a while after any of the following actions have occurred:

- When an EFT site is getting started (every time a site is started), it sets the Automate license and triggers an Automate service restart. Therefore, after a site is started, don't immediately open any Advanced Workflow tasks or Event Rules containing Advanced Workflows. It simply won't work, because it might be that the Automate service has not yet started, because this may take some time.
- Any changes to an EFT license, extending or deregistering the trial, or registering or deregistering the Advanced Workflows module will set the Automate license expiration dates accordingly and trigger Automate Service restart. Meaning that after any of those actions, you may have to wait until the Automate service gets started.
- Automate service is started or restarted in each of the cases described above and on EFT site start.

- Automate service is started or restarted when an Event rule is triggered, but for some reason Automate service shows it has stopped. Then EFT restarts it and then re-triggers the Event rule.

Automate Desktop Service Account

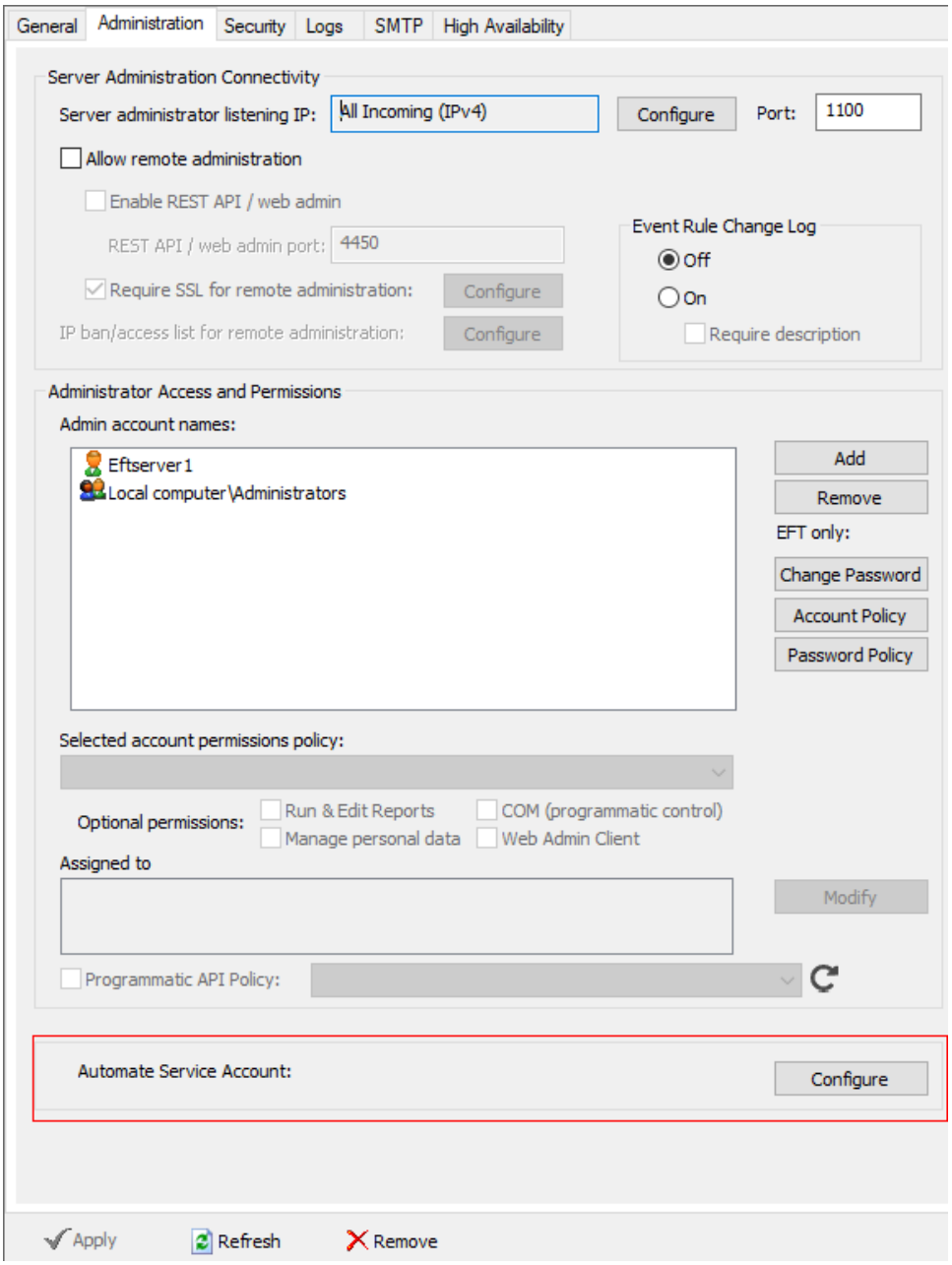
The Automate service enables tasks created using the Task Builder to trigger automatically on the computer. If this service is stopped, tasks will not be run at their scheduled times. The Task Builder requires a valid Windows user account to be enabled on the computer on which it is running to execute Advanced Workflows in event rules.

When the EFT administrator account or EFT service account has logged off of a Windows session, an *Automate service account* is needed to execute Advanced Workflows via event rules. EFT provides a way for the administrator to enter credentials for a Windows user account and persist them as part of EFT's configuration. The Automate Service account is enabled in the EFT administration interface on the **Server > Administration tab**.

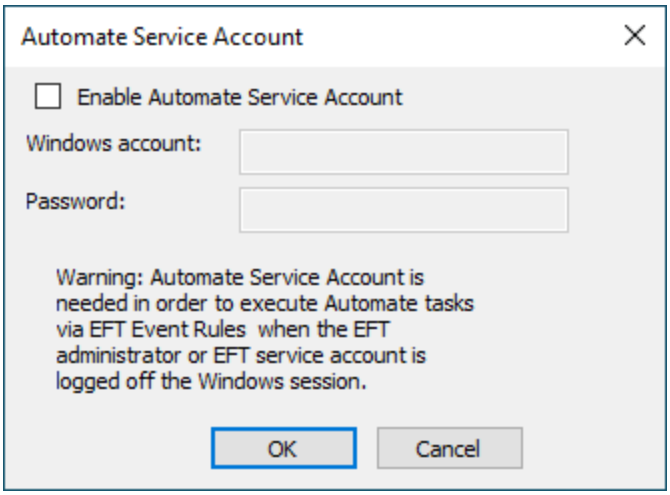
> **NOTE:** The AWM Dialog action requires user input. This is not desired since the EFT administration interface is not continuously monitored and a user prompt could delay the completion of event rules. You should enable the Automate Service Account to prevent the Dialog pop-up from occurring. Tasks will run in the background without popups.

To enable the Automate service account

1. At the bottom of the **Server > Administration** tab, next to **Automate Service Account**, click **Configure**.



The **Automate Service Account** dialog box appears.



2. Select the **Enable Automate Service Account** check box.
3. Provide the **Windows account** name and **Password** to be used for the Automate service account.
4. Click **OK** to close the dialog box and then click **Apply** to save the changes.

   Tasks will run in the background without popups.

Introduction to Advanced Workflows

The Advanced Workflow module adds additional automation capabilities to the EFT Event Rules system, allowing you to add scripting and variables to *Workflow Tasks*, then add these reusable Workflows to Event Rules. A Workflow is a series of steps that can perform file transfers, batch data processing, application testing, and so on, and are defined to run automatically when started by some event. The Advanced Workflow module is built on components of the Automate Desktop product from Fortra. EFT Event Rules use the Automate Task Builder and Actions, but not the Automate Task Administrator. Instead of the Automate Task Administrator, events and triggers are managed by the EFT Event Rules system.

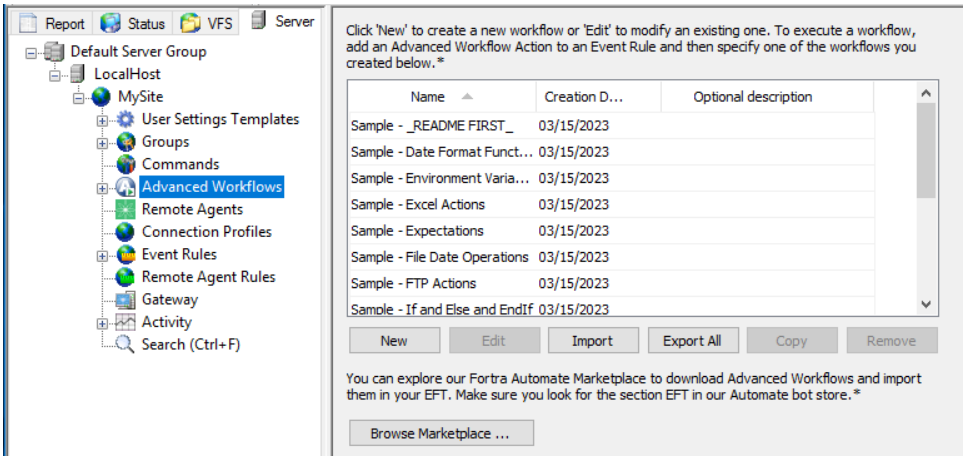
> **NOTE:** Both Automate Desktop Task Administrator and Automate Desktop Task Builder are installed with EFT. However, Automate 2024 will only work in EFT if the Advanced Workflow module is licensed in EFT (Help > About). The Automate Desktop service only starts when EFT triggers an event rule that uses Automate actions.

Some Automate actions are hidden by default from the list of available actions in the Advanced Workflow Task Builder. In many cases, these hidden actions are simply not applicable to the environment in which Advanced Workflows run, such as those geared around interactive desktop use cases or actions in which the Automate functionality is simply not carried over to the Advanced Workflow implementation in EFT (such as Twitter, Speech, and Multimedia actions). In other cases, such as for AS2 and Automate Schedule, they conflict with EFT modules. To find available Advanced Workflow module actions and descriptions that can be used in EFT Event Rules, refer to the Available Actions.

You can create your own workflows from the Advanced Workflows node in EFT or, in Automate 2024, get a few sample workflows from the Automate Connector Hub (Internet access required).

To go to the Automate Connector Hub

1. In the administration interface, connect to EFT and click the **Server** tab.
2. On the **Server** tab, click the **Advanced Workflows** node.
3. In the right pane, the **Advanced Workflows** tab appears.



4. In the right pane, click **Browse Marketplace**. The default browser opens to the Automation Connector Hub.
5. Scroll down on the page to browse all of the connectors.

Automation Connector Hub offers pre-built workflows, that you can download and import into EFT event rules. The Hub has numerous Automate tasks (Workflows) for applications from Active Directory to Zendesk. Click the link for the workflow that you are interested in. The linked page provides a description of the workflow and a link to download it. You can download multiple workflows at a time. There are more than 200 free workflows and about 15 "premium" workflows.

For details of specific Automate actions, please refer to the Automate Task Builder help inside EFT.

Workflow Concepts

The fundamental concept of Advanced Workflows is the automation of front and back-office business processes. Advanced Workflows is a software platform for building, managing, and launching automated tasks within EFT Event Rules. Tasks are developed via drag-and-drop without writing a single line of code. Simply drag together each action object in the sequence it should be executed.

Architecture

Advanced Workflows breaks down business processes into "visual" steps with the use of "actions." Multiple steps combined together are called a "task." Actions are used to instruct Advanced Workflows how to carry out a task. A "trigger" is a condition or event in Windows that essentially directs to execute a task automatically.

Task - The primary and most important object, a task is composed of a series of steps based on actions. Tasks can be designed to any of a very wide spectrum of business processes.

Actions - The available processes that can be used to build the steps to be carried out when a task is run. More than 300 actions are available from which to build tasks. These actions are assembled visually (without the use of code) via drag-and-drop in the Task Builder interface. Each action used to construct the task becomes a step. Upon execution, these steps are performed in a sequential order.

Components

The three main components are as follows:

Task Builder - Used to visually assemble actions into sequential task steps to be carried out. Additionally, the Task Builder is fully equipped with a wide variety of debugging tools to easily examine and troubleshoot problem tasks, eventually ensuring maximum reliability when the task is put into production.

EFT Event Rules - The central tool used to administer and customize all aspects of Advanced Workflows. EFT Event Rules organize tasks, connect to remote installations, and set task properties. In addition, it is used to administer global settings such as the default mail server, default logon properties, database connections, and much more.

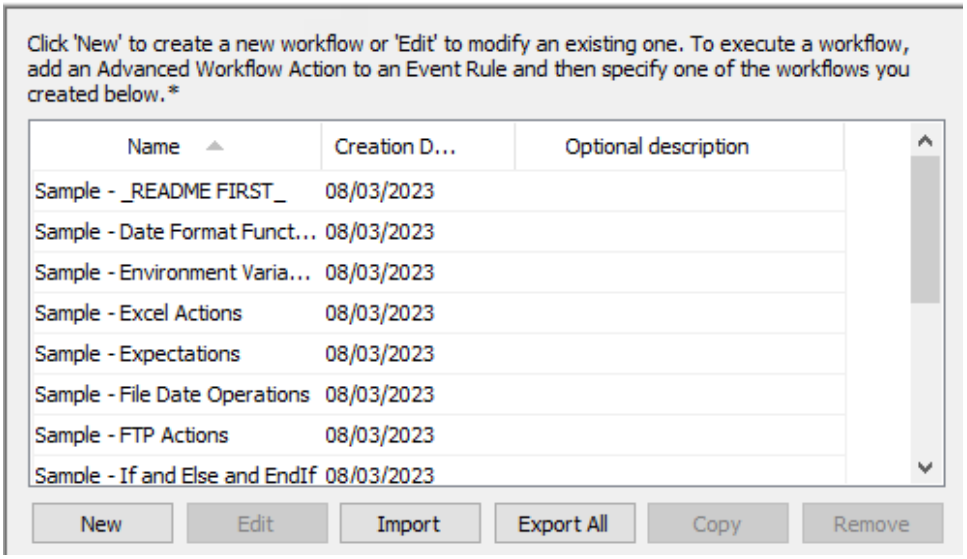
Advanced Workflow Action - Launches tasks according to the triggers attached to them.

The Workflow Task Builder Overview

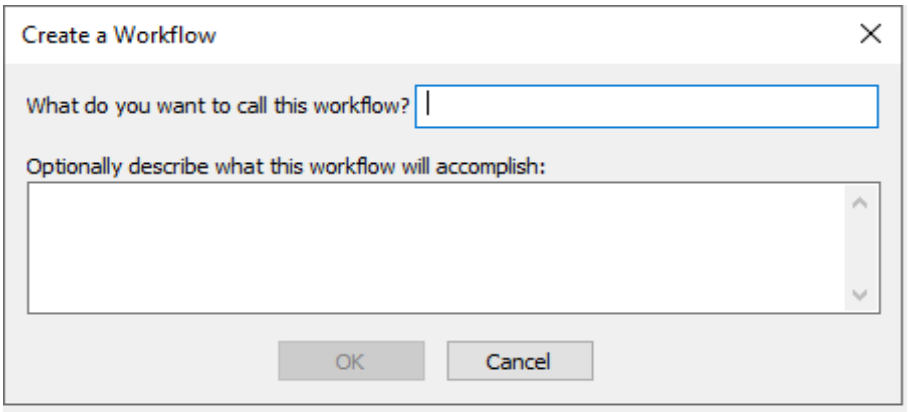
The Task Builder is used to create the workflow (Task) that you want to use as an event rule Action.

To open the Task Builder

1. In the administration interface, connect to EFT and click the **Server** tab.
2. On the **Server** tab, click the **Advanced Workflows** node.
3. In the right pane, the **Advanced Workflows** tab appears.

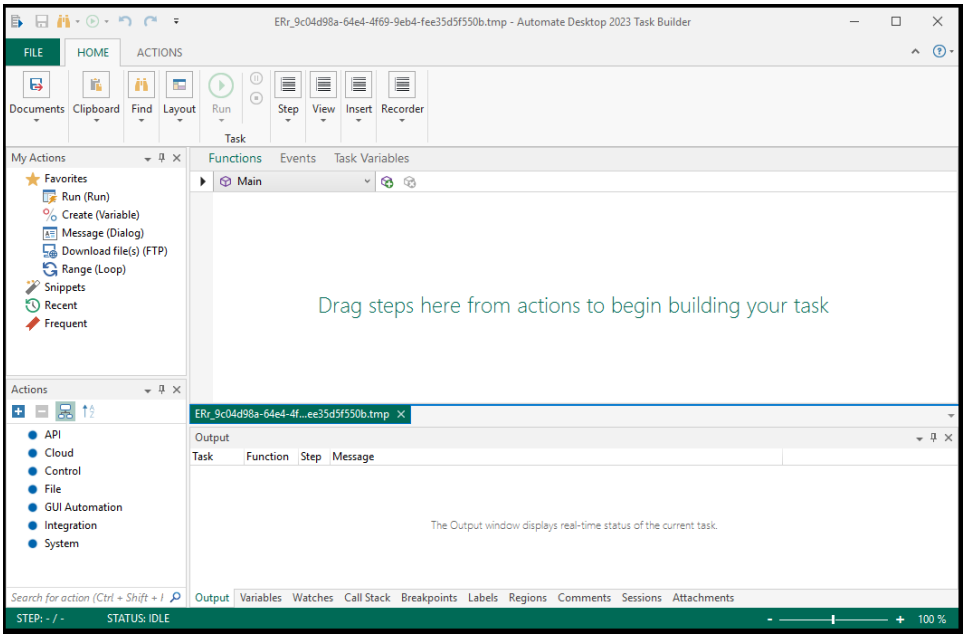


4. In the right pane, click **New**. The **Create a Workflow** dialog box appears.



5. In the **What do you want to call this workflow** box, specify a name for the Workflow, then click **OK**.

   The Workflow **Task Builder** appears.

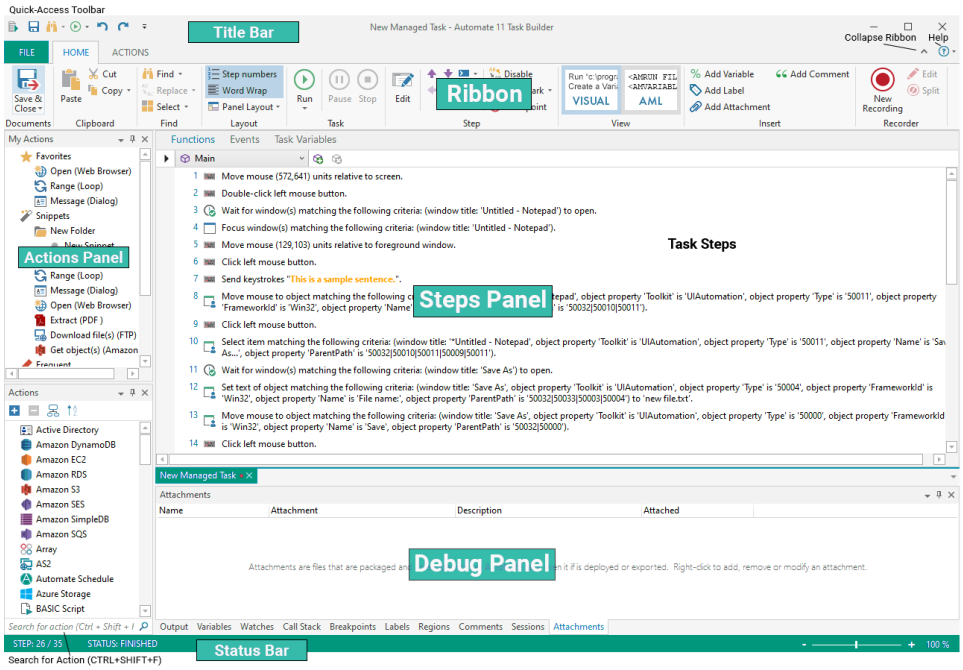


For details of how to create or edit workflows, refer to Creating Workflows for Use in an Event Rule.

Task Builder Interface

The Task Builder is an intuitive project designer used to visually assemble and test the activities that the task should carry out when it is executed (either manually or automatically by a trigger). Task steps are created by visually dragging-and-dropping Automate Desktop activities from an actions palette and placing them in the sequence desired for execution, eliminating the need to write code or create batch files. Additionally, the sequence of steps are described in plain English and presented in a way that clearly describes what they do.

The Task Builder includes a variety of testing and debugging features that displays verbose, real time data about a task, which aids the administrator in monitoring and debugging tasks as they are being constructed. This ensures that newly created tasks are functioning properly before they are put into production. For instance, the administrator can perform a series of “test runs” within the Task Builder and use a variety of intuitive debugging and diagnostic features to determine the cause of failing or problematic tasks.



The Task Builder interface is comprised of several panels or window panes. The table below lists each (in alphabetical order) along with a description.

Section	Description
My Actions	The **My Actions** panel saves a list of your favorite, recently used, and mostly used activities, and allows creation of snippets, which are copies of specific steps that contain predefined properties and settings.
Actions Panel	The **Actions panel** encompasses all of the actions available to construct the steps of a task.
Debug Panel	The **Debug** panel contains several views that are separated by tabs. Each view is comprised of a different debug tool. A different pop-up menu will appear for each view displaying the options available for that particular debug tool.
Ribbon	The **Ribbon** spans the top of the Task Builder, directly below the title bar. It is filled with graphical representations of control elements that are grouped by different functionality. Commands are organized in logical groups, which are collected together under operation related tabs.
Status Bar	The **Status Bar** is located at the bottom of the Task Builder and displays real-time progress of a running task through graphics and text.
Steps Panel	The **Steps** panel displays the actions and activities selected from the **Actions** Panel that will be carried out when the task executes.

To find available Advanced Workflow module actions and descriptions that can be used in EFT Event Rules, refer to the Actions panel in the Task Builder accessed from the EFT administration interface.

For details of specific Automate actions, please refer to the help in the Task Builder interface, or go open **..\Program Files\Globalscape\AutoMate\Help\Content\Task_Builder\Actions_Activities**.

Adding Task Steps

Once in the Task Builder, task construction can start by dragging the desired action from the Available Actions pane onto the Steps panel (as shown below). Doing so will open a properties dialog box for that action where parameters and specifications can be defined. Each type of action has its own set of properties. Specific actions may contain multiple activities to select from. Choosing the desired activity will populate the dialog with the proper properties and settings of that activity.

To add steps

1. In the Actions panel of the Task Builder, find the action/activity you want to add. You can find an action by clicking the plus signs (**+**) on any of the categorized folders, or by entering the name of the action or activity in the **Search for action** box at the foot of the Actions panel. When you use the **Search for action** box, the actions list is filtered as you type to include only those actions / activities that contain a match for the entered text (see Finding Actions for more details).
2. Drag the action / activity into the **Steps** panel (as shown below) or simply double-click the action / activity.

   The properties dialog box for that action opens.
3. Enter properties and click **OK**. The action / activity then becomes a step in your task. Repeat this process to add additional steps in sequence.
4. If you need to change the sequence of the steps, select the step you want to move and use the arrow buttons on the toolbar to move it up or down (for further information, see Editing Task Steps).
5. If you need to delete a step, right-click it, and select **Delete**.
6. When you are finished adding steps, click **Run** to test the task.
7. To save without closing Task Builder, from the Quick Access Toolbar (located above the ribbon by default), click the **Save** button or press CTRL+S.
8. To save and close Task Builder, from the ribbon, click the **Save and Close** button.

9. To save to a separate location, from the ribbon, select **File** > **Save Copy As** and navigate to the location in which to save your copy.

> **NOTE:** Unlicensed actions and activities will appear opaque in the Actions panel. Double-clicking, dragging or trying to add them to the Steps panel in any other way will generate an error.

Using add / insert option

- The Ribbon's **Actions** command group contains commands that can be used to add or insert a step.

To add an action as the last step

1. In the **Actions** panel, select the action / activity to add.
2. Click **Add Step** from the ribbon.

To insert an action in between steps

1. In the **Actions** panel, select the action / activity to insert.
2. In the **Steps** panel, select the step that you want the new action / activity to be inserted.
3. Click **Insert Step** from the ribbon. The new action will be inserted directly below the step you selected.

> **NOTE:** To add steps from the Recent, Frequent, Favorites, or Snippets folder, see My Actions Panel.

Editing Task Steps

Along with the ability to edit the Properties of individual task steps, many adjustments can be performed on any step once they are added to the Task Builder's Steps panel. You can move, indent, disable or delete one or more existing steps. You can perform cut, copy, and paste commands as well.

Commands are conveniently located on the Ribbon or by way of a context menu that appears when right-clicking a specific element. Certain ribbon commands become active only when items they apply to are selected, making it easier for you to distinguish relevant commands and avoid selection of unrelated ones.

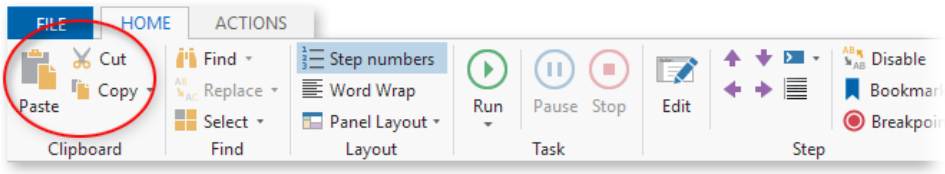
The Quick Access Toolbar contains **Undo** and **Redo** commands that apply to all edit operations allowing you to undo one or more incremental sequence of changes in case you make a mistake or redo them in case you change your mind.

Editing Step Properties

The properties of any task step can be easily viewed or modified once it is added to Task Builder's **Steps** panel. For convenience, step properties can be accessed in various ways.

To view or modify the properties of a step

1. In the **Steps** panel of Task Builder, select the step that contains the properties you wish to modify and double-click on the step.
2. Perform the desired changes in the properties dialog that appears, and then click **OK** to save changes.



Moving Steps

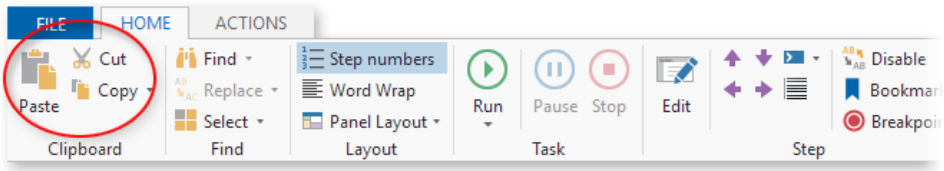
Steps can be moved freely within the Steps panel. You can modify the sequence at which steps execute by moving them above or below other steps. Task Builder supports movement of multiple sequenced steps (for example, steps 3, 4, and 5) as well as multiple non-sequenced steps (for example, steps 3, 6 and 8). For convenience, such operations can be performed in various ways.

To move steps using the Move Up or Move Down commands

1. In the **Steps** panel of the Task Builder, select the steps you want to move up or down the step sequence. To select more than one step, hold down CTRL during selection.
2. From the ribbon, click the **Move Up** or **Move Down** arrow. Each click is equivalent to a single line of movement.

To move steps using Cut and Paste commands

1. In the **Steps** panel of Task Builder, select the steps you want to move up or down the step sequence. To select more than one step, hold down CTRL during selection.
2. From the ribbon, click **Cut** (you can also use CTRL+X) .
3. Select a desired location in the **Steps** panel and click **Paste** from the ribbon (you can also use CTRL+V).



To move steps using Drag & Drop actions

1. In the **Steps** panel of Task Builder, select the steps you want to move up or down the step sequence. To select specific steps or a block of steps, hold down CTRL during selection.
2. Hold down the left mouse button while dragging the highlighted steps to the desired location, then release the mouse button to complete the operation.

Indenting Steps

You can increase or decrease indentation of one or more steps as a way to organize your steps. Task Builder supports indentation of multiple sequenced steps (for example, steps 3, 4 and 5) as well as multiple non-sequenced steps (for example, steps 3, 6 and 8).

To increase or decrease indentation

1. In the **Steps** panel of Task Builder, select the steps to increase or decrease their current indentation. To select a block of sequenced steps or specific non-sequenced steps, hold down CTRL during selection.
2. To increase indentation, click the arrow pointing right in the ribbon's **Step** command group (you can also use CTRL+ right arrow key). To decrease indentation on an indented step, click the arrow pointing left (you can also use CTRL + left arrow key).

> **NOTE:** Left justification of an indented step is based on the parent step (if relevant) and the default indentation settings applied by way of Task Builder **Options > Formatting**.

To reset indentation

1. In the **Steps** panel of the Task Builder, select the steps that should be reverted back to their original indentation level. To select a block of sequenced steps or specific non-sequenced steps, hold down CTRL during selection.
2. From the ribbon, click the **Reset Indent**. Hold down the left mouse button while dragging the highlighted steps to the desired location, then release the mouse button to complete the operation.

You can reset one or more steps to their default indentation level. From the ribbon's **Step** command group, click the **Format Steps** command. This enables or disables auto-formatting of steps. When enabled, blocks of steps contained within a conditional (If) statement or Loop process are formatted (indented) to improve distinction (this command is enabled by default).

Disabling, Enabling, or Deleting Task Steps

Steps of a task can be disabled or re-enabled to use or test different versions of the same task. The **Disable** button on the ribbon or context menu acts as a toggle to disable an enabled step or re-enable a disabled step. When a step is disabled, it is grayed out by default in the **Steps** panel and it is treated as a comment and ignored when the task runs.

To disable or enable a task step

1. In the **Steps** panel of the Task Builder, select the steps you want to disable. To select more than one step, hold down CTRL during selection.
2. Click the **Disable** button on the ribbon under the Home tab (you can also right-click on the steps, and then click **Disable**).

> **NOTE:** To re-enable, follow the same instructions.

To delete a task step

1. In the **Steps** panel, select the step or steps you wish to delete. To select more than one step, hold down CTRL during selection.
2. Right-click on your selection, and then click **Delete** (you can also use the Delete key).

Step Numbers & Word Wrap

By default, the **Steps** panel displays numbers to represent each step in the task are displayed for each step in the Task Builder's **Steps** panel along with an icon designating the type of action used. Also the text (or AML code) that appears in this panel is word wrapped by default allowing all text to be visible in the window regardless of its size. Nonetheless, these options can be customized according to each user's preference.

To turn Word Wrap on or off

1. From the Ribbon, click **Home**.
2. From the **Layout** command group, click the **Word Wrap** button to toggle Word Wrap on/off.

To disable or enable step numbers

1. From the Ribbon, click **Home**.
2. From the **Layout** command group, click **Step numbers** to toggle step numbers on/off (also shown below).

Switching from Visual to the AML View

The Task Builder offers two ways in which to view the steps of a task. The default appearance displays a plain-English, easy to comprehend text description of each step. They can also be displayed in AML (Automate Desktop Markup Language) format, which is the internal program language encompassed in Automate Desktop. When set to this view, those who understand AML can edit their code directly from the Task Builder's Steps panel or copy the code onto an external text editor such as Notepad.

To switch from Plain Text to AML or AML to Plain Text

1. From the Ribbon, click **Home**.
2. In the **View** command group, select **AML** to switch to AML code view or **Visual** to switch to plain text view.

To edit code in the AML view

1. In the **Steps** panel, select the step to edit.
2. Click the step again to make the code editable.
3. Make the desired changes.

Undoing and Redoing Changes

To enable task development to be as simple and painless as possible, an **Undo** and **Redo** command is included on the Quick Access Toolbar. Undo reverses the last action you performed, and Redo undoes the last Undo action, in case you change your mind. Task Builder keeps track of all previously performed actions, therefore, you can reverse the actions you performed one action at a time, in reverse consecutive order, up to and including the first action.

To undo changes

- Click the **Undo** command on the Quick Access Toolbar or press CTRL + Z. Undo reverses the last editing action. If you delete one or more steps, you can use **Undo** to restore them and all their properties. You can perform as many Undo actions as you like.

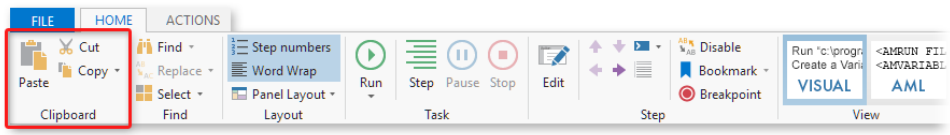
To redo a change after an undo

- Click the **Redo** command on the Quick Access Toolbar or press CTRL + Y. This restores the last action you performed if no other actions have occurred since the last Undo. The **Redo** button is normally inactive and becomes active only if an **Undo** is performed.

Copying and Reusing Task Steps

The reuse of code is a common technique In computer programming, used to save time, reduce redundancy and eliminate errors by applying the same segment of code in multiple applications. The Task Builder plays a similar role by allowing task steps to be copied and re-used within the same task or across multiple tasks. This article describes various methods to perform this operation.

The **Home** tab of the ribbon contains clipboard commands that allow you to reuse steps within a task by copying one or more steps and pasting them onto another section of the task. Because more than one instance of Task Builder can be opened simultaneously, code reuse can be performed across multiple tasks as well. The **Copy** command, in effect, copies the .AML code associated to each task step. This code can be pasted onto a specific step on the same task or another task using the **Paste** command. Alternatively, a **Copy > Description** command is available, allowing the textual description of each step to be copied and pasted onto another application outside of Task Builder, such as an email message or text file.



To copy and paste steps in the same task

1. Select the task you want to work with to open Task Builder.
2. From the Steps panel, select the steps you wish to copy. To select a group of steps, hold down CTRL during selection. Use CTRL+A to select all.
3. On the ribbon, select the **Home** tab and click the **Copy** button from the Clipboard command group. Or, right-click the highlighted steps and select **Copy** from the context menu.
4. From the Steps panel, select the step you want to place the previously copied steps. Then, on the ribbon, select the **Home** tab and then click the **Paste** button from the Clipboard command group.

The copied steps are pasted precisely at the step you selected. Existing steps are moved directly below the pasted steps.

To copy steps from one task to another existing task

1. Select the task you want to copy steps from in order to open the first instance of Task Builder.
2. Select the task you want steps to be pasted onto in order to open the second instance of Task Builder.
3. From the Steps panel in the first instance of Task Builder, select the steps you wish to copy . To select a group of steps, hold down CTRL during selection. Use CTRL+A to select all.
4. On the ribbon, select the **Home** tab and then click the **Copy** button from the Clipboard command group.
5. From the Steps panel of the second instance of Task Builder (the one you wish to paste steps onto), on the ribbon, select the **Home** tab and then click the **Paste** button from the Clipboard command group.

The copied steps are pasted precisely at the step you selected. Existing steps are moved directly below the pasted steps.

To copy steps from one task to a new task

1. Create and edit a new task to open an instance of Task Builder.
2. Select the existing task you want to copy steps from to open another instance of Task Builder.
3. From the Task Builder's Steps panel of the existing task, select the steps you wish to copy . To select a group of steps, hold down CTRL during selection. Use CTRL+A to select all.
4. On the ribbon, select the **Home** tab and click the **Copy** button from the Clipboard command group.
5. Select the Task Builder instance associated to the new task.
6. Right-click anywhere inside the empty Steps panel and select **Paste** from the context menu. The copied steps are pasted directly onto the Steps panel of the new task.

To copy and paste step descriptions

1. From the Steps panel of Task Builder, select the steps in which to copy the description from. To select more than one step, hold down CTRL during selection or use CTRL+A to select all steps.
2. From the ribbon interface, select **Copy > Description** from the **Clipboard** command group or right-click and select **Copy > Description** from the context menu.
3. The text can be pasted into any Automate Desktop field or text-box where plain text is accepted. You can also paste copied steps or step descriptions in other documents such as email messages or text files.

NOTE:

- When **Copy** > **Description** is selected, the content of the clipboard becomes populated with the “Visual View” text-only description that is normally displayed in the Steps panel and does not contain any AML code (even if the Steps panel is currently set to be viewed in AML format).
- To copy the AML code of a particular step instead of the text description, select **Copy** from the ribbon interface or right-click the step and select **Copy** from the context menu (not **Copy** > **Description**). Doing so will allow you to paste the AML code into a document, text file or any field in Automate Desktop that accepts text.
- Selecting **Copy** will also allow you to paste the content as a step in the Steps panel.

Steps of an existing task can be re-used in a newly created task by selecting the **Save Copy As** command. This is an ideal method for copying all of the steps of an existing task onto a new task.

To save a copy of a task

1. Open the task you wish to copy steps from by doing one of the following:
    - From Task Administrator, right-click the task and select **Steps**. The Task Builder appears for this task.
    - Select the task from the Repository section of the Server Management Console or Workflow Designer. A separate instance of the Task Builder appears for each task.
    - If a new (unmanaged) task was initially created by directly opening the Task Builder from the Start menu, select **Open** from the Application menu and navigate to the task (.AML) file you wish to open and click the **Open** button. In this case, a separate tab appears in the Steps pane of the Task Builder, each signifying an open task.
2. From Task Builder's ribbon interface, select **File** > **Save copy as**.
3. Navigate to the desired location, enter a name for the new task, and click **Save**.

Using Snippets

Another way of using a group of steps within a specific task is to save the steps as Snippets. Snippets provide an easy way to implement frequently used code into a task. They are particularly useful to those who use specific activities that contain commonly used properties or settings. Instead of re-entering the properties for a specific activity or series of activities every time they're added as task steps, users can save existing steps as a snippet and simply drag and drop the snippet wherever it is needed in any task.

Finding and Replacing Text in Task Steps

Task Builder encompasses a comprehensive Find and Replace feature that enables you to search for every occurrence of a particular word or phrase in one or more open tasks, and optionally, replace them with the specified text. You can scope searches to the current task, current function, selected steps or all open tasks. Options and rules that you set can restrict the results to those that meet the criteria that you specify. For example, you can specify what direction you want to search or require that only text that matches the case of the text you entered should be found.

The **Find** and **Replace** controls are located in Task Builder's ribbon under the **Home** tab. To determine related hot-keys, hover your cursor over each control.

Finding text

The Find dialog searches through the AML code of the selected step or function, current task, or all open tasks (depending on the search options you choose). When finding one instance of text at a time, clicking **Find** highlights the next instance of text found.

To find text in one or more steps/tasks

1. In the Steps panel, click the step from which you want to start the search or specific steps you want to include in the search. To select more than one step, hold down CTRL during selection.
2. From Task Builder's ribbon, select the **Home** tab and click **Find** or press **CTRL + F**. The Find dialog appears.
3. In the Find dialog, do the following:

Use this...	To do this...
Look in	Select the scope of the search: • **Current task** - Search the AML of the current task. • **Current function** - Search the current function. • **Selected steps** - Search only the selected steps. • **All open tasks** - Search all tasks open in the Task Builder. The Task Builder application (AMTB.exe) must be opened directly in order for it to display multiple tasks. If it is opened by way of a managed task, then
Find what	Type the text you want to search for.
Match case	Find text only if it matches the case of the text you typed.
Match whole words	Find text only if it matches the whole word you typed.
Regular expression	Use regular expressions (notations for patterns of text) rather than the literal character. For more information, see **Regular expressions for finding text** below.
Search up	Searches will be performed from the bottom to the top.
Wrap searches	Automate Desktop will search for the specified text beginning at the point in the document where you are currently positioned, and will continue past the end, to the beginning of the document back to your current position. In other words, wrap search will search the entire document irrespective of where you may be positioned within it

4. Click **Find** . The found text (if any) is highlighted in the task.
5. Repeat clicking **Find** to find additional instances of the text.
6. Click **Close** when finished.

> **NOTE:** You can close the **Find** dialog and still find the next instance by clicking **Find Next** on the ribbon.

Replacing text

You can automatically replace text using the **Replace** dialog. This dialog has the same functionality as **Find** but allows you to make text substitutions in the current function, steps or tasks as shown below.

To replace text in a task

1. From the Task Builder ribbon, select the **Home** tab and click **Replace**. This opens the **Replace** dialog
2. In the **Replace** dialog, do one of the following:

Use this...	To do this...
Look in	Select the scope of the search: • **Current task** - Search and replace text from the AML of the current task. • **Current function** - Search and replace text found in the current function. • **Selected steps** - Search and replace text from the selected task steps. • **All open tasks** - Search and replace text within all open tasks in the Task Builder. The Task Builder application (AMTB.exe) must be opened directly in order for it to display multiple tasks. If it is opened by way of a managed task, then
Find what	Type the text you want to search for.
Replace with	Type the text you want to replace the original text with.
Match case	Find text only if it matches the case of the text you typed.
Match whole words	Find text only if it matches the whole word you typed.
Regular expression	Use regular expressions (notations for patterns of text) rather than the literal character. For more information, see **Regular expressions for finding text** below.

Use this...	To do this...
Search up	Searches will be performed from the bottom to the top.

3. Select one of the following:
    - To find one instance of the specified text at a time, click **Find Next**. The next instance is selected in the Steps panel in AML view. Repeat clicking **Find Next** to find additional instances of the text.
    - To replace the current text instance with the new text, click **Replace**.
    - To replace the ensuing text instance with the new text, click **Replace Next**. Repeat clicking **Replace Next** to replace additional instances with the new text.
    - To replace all instances of the specified text with the new text at the same time, click **Replace All**.
4. Click **Close** when finished.

Regular expressions for finding text

You can perform sophisticated find and replace operations in the Task Builder by using regular expressions. Regular expressions are useful when you do not know the exact text or code you are looking for, or when you are looking for all occurrences of strings of text or code with one or more similarities.

A regular expression is a pattern of text that describes one or more variations of text or code that you want to find. A regular expression consists of specific characters, such as the letters **a** through **z** and special characters that describe the pattern of text, such as an asterisk **(*)**. For example, to find all variations of the word **page** in your task, you can enter **page*.** This finds all instances of **page**, **pages**, **pager**, and any other words that begin with **page** in your task.

When you use regular expressions in your searches, there are specific rules that control which combination of characters perform specific matches. Each regular expression or combination of regular expressions is referred to as syntax. You can use multiple regular

expressions in one syntax to precisely target your search. The following table displays a list of common expression syntax and their description.

Syntax	Expression Description
.	Any character acts as a wild card to match any single printing or non-printing character with the exception of the newline (\n)character. For example, the regular expression c.t matches the strings cat, c t, cot, but not cost. In this example, the period (.) is a wild card for a single character. It appears between the letters 'c' and 't', so any single character between the characters 'c' and 't' will match the expression—even if it is a space.
*	Maximal — zero or more matches zero or more occurrences of a character that precede the expression, matching as many characters as possible. The regular expression .* matches zero or more occurrences of one character. For example, the regular expression b.*k matches book, back, black, blank, and buck. In this example, we combine the period (.) with the asterisk (*) to make one syntax. The period (.) appears immediately before the asterisk (*) expression. The asterisk (*) matches zero or more occurrences of any character between 'b and 'k'. The period (.) acts as a wild card for the characters between 'b' and 'k'. In this example, it means that any character between 'b' and 'k' can be repeated.

Syntax	Expression Description
+	Maximal — one or more matches one or more occurrences of a character that precede the expression, matching as many characters as possible. The regular expression .+ matches one or more occurrence of one character.For example, the regular expression bo+. matches bob, book, and boot. In this example, we combine the period (.) with the plus sign (+) to make one syntax. The period (.) appears immediately after the plus sign (+) expression. The plus sign (+) matches one or more occurrences of the letter 'o'. The period (.) acts as a wild card for the last character of each word, which, in this example are 'b', 'k', and 't'.
@	Minimal—zero or more matches zero or more occurrences of a character that precede the expression, matching as few characters as possible. The regular expression .@ means match zero or more occurrences of one character. For example, the regular expression a.@x matches 'abx' within '
#	Minimal—one or more matches one or more occurrences of a character that precede the expression, matching as few characters as possible. For example, the regular expression si.#er matches 'sicker' or 'silkier'. In this example, we combine the period (.) with the sharp sign (#) to make one syntax. The period (.) appears immediately before the sharp sign (#) expression. The sharp sign (#) matches one or more occurrences of any character between 'si' and '
[ ]	Set of characters matches any one of the characters within the brackets ([ ]). You can specify ranges of characters by using a hyphen (-), as in [a-z]. Examples: • The regular expression c[aou]t matches cat, cot, and cut, but not • The regular expression [0-9] means match any digit. You can specify multiple ranges of letters as well. • The regular expression [A-Za-z] means match all upper and lower case letters.

Syntax	Expression Description
^	Beginning of line anchors the match to the beginning of a line. For example, the regular expression ^When in matches the any string that begins with "When in" and that also appears at the beginning of a line, such as "When in the course of human events" or "When in town, call me". Whereas, this regular expression does not match "What and when in the course of human events" if it appears at the beginning of a line.
$	End of line anchors the match to the end of a line. For example, the regular expression professional$ matches the end of the string "He is a professional" but not the string "They are a group of professionals".
^^	Beginning of file anchors the match to the beginning of a file. Works only when searching for text in source code or in text files. For example, to match the first HTML tag at the beginning of a file, use the following regular expression: ^^
$$	End of file anchors the match to the end of a file. Works only when searching for text in source code or in text files. For example, to match the last HTML tag at the end of a file (with no spaces following the tag), use the following regular expression: $$
|	Or indicates a choice between two items, thereby matching the expression before or after the OR symbol (|). For example, the regular expression (him|her) matches the following occurrences it belongs to him or it belongs to her but it does not match the line it belongs to them.
\	Escape special character matches the character following the back slash (\ ). This allows you to find characters that are used in the regular expression syntax, such as a left curly brace ({) or a caret (^) or some other special character. For example, you can use \$ to match the dollar sign character ($) rather than implementing the regular expression to 'anchor to end of a line'. Similarly, you can use the expression \. to match the period (.) character rather than match any single character, which is what the period (.) regular expression does.

Syntax	Expression Description
{}	Tagged expression tags the text matched by the enclosed expression. You can match another occurrence of the tagged text in a Find expression or insert the tagged text in a Replace expression using \N. For example, suppose you are looking to find two duplicate, consecutive words. To search, use the following expression: {.#} \1 With the assumption that the consecutive words are separated by a single space, you'll want to add a space between the right curly brace (}) and the back slash (\ ). In this example, we combine the sharp sign (#) and the period (.) with the curly braces ({}) to make one syntax. In this expression, .# represents any consecutive characters. Since this portion of the expression is surrounded by curly braces ({}), the consecutive characters will be tagged and can be referred to as \1. This expression will find any consecutive characters followed by a space, followed by those exact same consecutive characters.

Syntax	Expression Description
\N	Nth tagged expression in a Find expression, \N matches the text matched by the Nth tagged expression, where N is a number from 1 to 9. In a Replace expression, \N inserts the text matched by the Nth tagged expression where N is a number from 1 to 9. \0 inserts the text matched by the entire Find expression. For example, suppose you want to find two duplicate, consecutive words and replace them with a single word. To search, use the following expression: {.#} \l With the assumption that the consecutive words are separated by a single space, you'll want to add a space between the right curly brace (}) and the back slash (\ ). In this example, we combined the sharp sign (#) and the period (.) with the curly braces ({}) to make one syntax. To replace, use the following expression: \l \1 represents what was found in the first pair of curly braces in the find string. By using \1 in the replace action, you essentially replace the duplicate, consecutive words with a single copy of the word.
( )	Group expression marks the beginning and end of a sub expression. A sub expression is a regular expression that you enclose in parenthesis ( ), such as the expression that follows: (ha)+ In this example, we combine the plus sign (+) with the parenthesis ( ) group expression to make one syntax. The sub expression is (ha) because it is encapsulated within the parenthesis ( ). When you add the plus sign (+), the expression enables you to find repeating pairs of letters. The plus sign (+) represents one or more occurrences of 'ha'. This expression matches the following occurrences 'haha' and '

Syntax	Expression Description
~x	Prevent match prevents a match when x appears at this point in the expression. For example, the regular expression real~(ity) matches the real in realty and really, but prevents the match to real in reality.
\n	Line break matches a new line in Code view, or a <br inDesignview. The syntax (\n), is a shorthand approach to enable you to match all line breaks.
\t	Tab matches a single tab character. For example, if you want to find all single tabbed characters at the beginning of a line, the regular expression would look like the following: ^\t+ In this example, we combine the caret (^) and the plus sign (+) with the tab (\t) to make one syntax. The caret (^) that precedes the single tab character expression, anchors the match to all tabbed characters at the beginning of the line. The plus sign (+) represents the matching of one or more tab characters.
[^]	Any one character not in the set matches any character that is not in the set of characters that follows the caret (^). For example, to match any character except those in the range, use the caret (^) as the first character after the opening bracket. The expression [^269A-Z] will match any characters except 2, 6, 9, and any upper case alphabetical characters.
n	Repeat expression matches n occurrences of the expression that precedes the caret (^). For example, with n equaling 4, the expression [0-9]^4 matches any 4-digit sequence. In this example, we combine the set of characters ([ ]) syntax with the repeat (^n) syntax to demonstrate a more realistic use of regular expressions.

Syntax	Expression Description
:a	Alphanumeric character matches the expression [a-zA-Z0-9]. You can use the following expression: [a-zA-Z0-9] to match one occurrence of a letter (upper case or lower case) or number. Also known as alphanumeric occurrences. You can use the shorthand expression :a for all instances of [a-zA-Z0-9].
:b	White space matches any white spaces in code or text. For example, to match a single white space character at the beginning of a line, use the following regular expression:^:b
:c	Alphabetic character matches the expression [a-zA-Z]When you use this expression, it enables you to match all upper or lower case letters. You can use the shorthand expression :c for all instances of [a-zA-Z].
:d	Decimal digit matches the expression[0-9]This expression enables you to match any digit. For example, suppose you want to find a social security number in a text file. The format for U.S. social security numbers is 999-99-9999. :d^3-:d^2-:d^4 or, by using [0-9], the same resulting expression:[0-9]^3-[0-9]^2-[0-9]^4] You can use the shorthand expression :d for all instances of [0-9].

Syntax	Expression Description
:h	Hexadecimal digit matches the expression [0-9a-fA-F]+ Use a this expression when you want to match a hexadecimal combination of any upper or lower case letters between 'A' and 'F', and any numbers. For example, suppose the pages in your website have multiple different background colors and you want to change the color of those pages to black, such as 000000. However, you do not know what the hexadecimal numbers are for the existing colors. Use the following regular expression to find all existing hexadecimal numbers: \#:h You could use [0-9a-fA-F] to search, but in this example we combine the back slash (\) and the sharp sign (#) with the hexadecimal digit (:h) syntax. \# matches a non-expression sharp sign (#) and :h matches any sequence of hexadecimal characters. To replace the existing hexadecimal numbers, type the hexadecimal number of the background color that you want: 000000
:i	Identifier matches the expression [a-zA-Z_$][a-zA-Z0-9_$]* When working with code, if you want to match all program identifiers, you can use the shorthand expression :i to replace having to type the lengthy expression above.
:n	Rational number matches the expression ([0-9]+\.[0-9]*)|([0-9]*\.[0-9]+)|([0-9]+) If you want to match all whole numbers that contain a decimal point, you can use the shorthand expression :n to replace having to type the lengthy expression above.

Syntax	Expression Description
:q	Quoted string matches the expression ("[~"]*")|('[~']*') If you want to match all quotes surrounded by quotation marks, you can use the shorthand expression :q to replace having to type the lengthy expression above.
:w	Alphabetic string matches the expression [a-zA-Z]+ This syntax is a shorthand approach to enable you to match one or more alphabetical characters, either lower case or upper case.
:z	Decimal integer matches the expression [0-9]+ This syntax is a shorthand approach to enable you match any number from zero or more.

Task Builder File Menu

The File menu comprises a variety of typical commands, such as **New**, **Open**, **Save**, **Print**, **Send**, and **Close**. It also contains an **Options** command which allows you to view or edit various Task Builder preferences, including formatting options, default fonts, color schemes and debugging alternatives. For convenience, the File menu lists the most recently opened tasks so that you don't have to search your computer for a task you previously opened or frequently work on.

The File menu can be accessed by clicking the **File** tab located in the upper-left corner of the ribbon.

File menu items

The following table provides a description for each command found when clicking **Application** along with their associated controls and keyboard hot-keys (if applicable).

Command	Description	Hot-Key
Save	Saves the current task. If the task being saved is a managed task, it will be saved to its present location. If saving an unmanaged task (that is, created using the **New** option), a standard Windows Explorer dialog appears allowing you to select the folder in which to save the task (.AML) file.	CTRL + S
Save Copy As	Saves a copy of the task file or saves the task file in another format. A standard Windows Explorer dialog appears allowing you to navigate to the location in which to save the task (.AML) file.	SHIFT + CTRL + S
Print	Prints the steps of the current task as they appear in the Steps pane.	---
Send	Emails the current task to the specified recipient. The available send options are: • **Mail Recipient** - Sends the displayed task in the body of an email. • **Mail Recipient (As Attachment) -** Sends the displayed task as an email attachment. • **Mail All to Recipient (As Attachment) -**Sends all open tasks as email attachments.	---
Tools		
Close	Closes the current task file. If changes were made, a dialog will appear prompting whether to save the changes. Click **Yes** to proceed.	---
Close All	Closes all currently opened tasks. If changes were made to one or more tasks, a dialog will appear prompting whether to save the changes. Click **Yes** to proceed.	---

Command	Description	Hot-Key
Options	Allows access to available Task Builder Options. The table below lists available options.	---
Exit	Closes the Task Builder.	---

Available Task Builder Options

Below lists the various categories of Task Builder options in alphabetical order, with links to the topics covering each:

Option	Description
General Options	Contains options that determine certain behavioral characteristics of Task Builder. Also determines the default managed task location.
Formatting Options	Contains options that deal with how steps should be formatted in the Steps pane.
Fonts Options	Contains options to modify the font name, size and style to be used for the Visual view and the AML view.
Color Options	Contains options that determine the color to be used for the Task Builder interface or allows adjustments to be made to the current color scheme.
Layout Options	Resets the Task Builder panels back to the default settings.
Debugger Options	Contains options that affect Task Builder's behavior while running and debugging tasks.
Toolbar Options	Contains options that determine the look and feel of Automate Desktop's debugger toolbar.

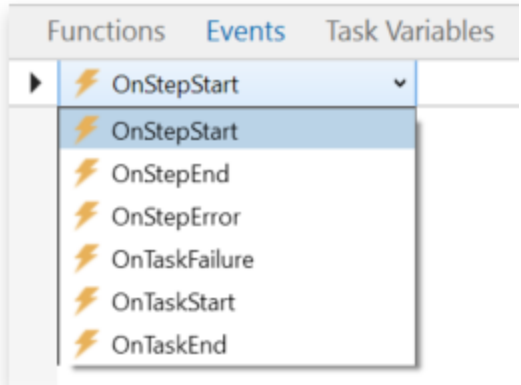
Task Events

Task events along with task functions are core features in programming that make complex, high level automation routines more comprehensible. They enable the flow of a program to be determined by specific events that take place, which in turn, triggers execution of a callback function.

The Task Builder is capable of raising task events as it performs the steps necessary to complete a task. This capability allows specific event-driven information to be returned when a task starts, when it completes (either successfully or unsuccessfully) or during various other stages in between (for example, when a task step starts, ends or generates an error). Tasks become more robust when task events are implemented. This is because events make a task more intelligent with their ability to monitor execution status and provide notification when a status has changed.

Task events are managed through a drop-down menu that is accessible from the Steps panel of Task Builder.

NOTE: Task Events are additional features aimed for advanced users. You can build tasks without the use of Task Events and may choose to use these features at your own pace.



To set a task event

1. From the Task Builder's Steps panel, select **Events**.
2. Click the drop-down combo box to display a list of pre-defined task events (see the

table below for more details).

3. Select the desired task event.

Parameters

The following table describes available task event parameters:

Property	Type	Description
Name	Text	The name of the parameter.
Return type	Text	The return value type. Task event value types are always in the form of a variable.
Is Optional	TRUE/ FALSE	Specifies whether the parameter is optional (TRUE) or required (FALSE). This option is always marked FALSE for task events.
Default value	Text	The task event's default value.
Description	Text	A description of the parameter.

Available Events

The Task Builder provides six task events in which you can subscribe (also known as pre-defined task functions). The following table lists the available task events along with their description. For additional information regarding a specific task event, click the event's associated link.

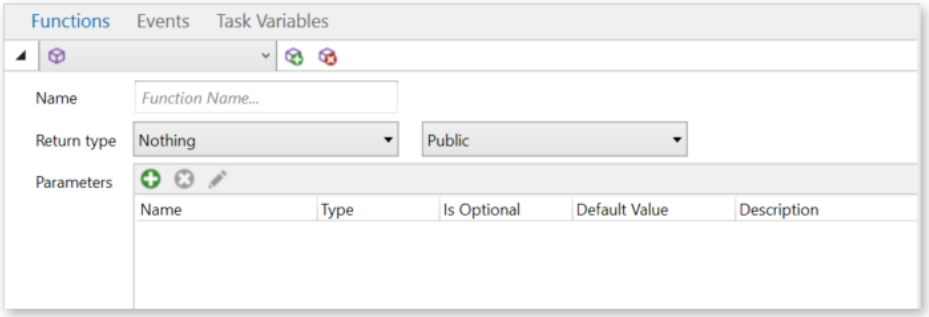
Event	Description
OnStepStart	This event is fired each time a step executes.
OnStepEnd	This event occurs each time a step is completed.
OnStepError	This event occurs each time a task step generates an error.
OnTaskFailure	This event is fired each time a task step causes an error. Depending on how the step is configured, the task may either continue after the error occurs or may be aborted. In the latter case, the **OnTaskEnd** event is fired immediately after.

Event	Description
OnTaskStart	This event is fired when a task is started. This is normally the first event fired when task execution starts.
OnTaskEnd	This event is fired when a task ends.

Task Functions

In programming, functions allow the structuring of programs in segments of code to enable performance of individual tasks. A function has input and output, and contains instructions used to create the output from its input (that is, performs a calculation on the input and returns an output). The same holds true for task functions in the Task Builder. Each task function performs an operation and optionally returns a value. The output value can be private, or it can be made public and stored in a variable to be called upon by as many steps as required by the task.

Task functions are created and managed through a drop-down interface accessible from the Steps Panel of Task Builder.



NOTE: Task Functions are supplemental features aimed for advanced users. You can build tasks without the use of Task Functions and may choose to use these features at your own pace.

Scope and Accessibility

The power of task functions comes from their ability to provide variable modularity and protection and their capability to arrange a collection of task steps into a logical structure.

This is accomplished through two distinct yet complimentary standards; scope and accessibility. Scope and accessibility are primary building blocks that allows you to construct a solid foundation and enable a rich object-oriented approach to tasks. They provide distinct capabilities and serve different but complimentary roles necessary to distinguish the benefits of task functions.

Scope

A scope in any programming is a region of the program where a defined variable can have its existence and beyond that variable can not be accessed. In Automate Desktop, scope limits a task function's visibility within a task. Without proper scoping, some of the primary advantages of function-oriented design would not be possible, including information-protection, modularity, maintainability and recursion. Scoping helps take large, unruly tasks and enforce logical restrictions to their structure to provide greater readability and maintainability. Scoping also helps avoid unintentional and confusing data changes leading to easier debugging as well as construction of cleaner and more reliable tasks.

Accessibility

Accessibility provides the fundamentals of information-protection, encapsulation and interfacing, all of which are essential for an object-oriented approach to a language. It makes tasks more portable, manageable and documentable by providing outside access only to those parts of the task that are meant to be used, while providing the user full flexibility of functions. Task functions exist at the root of a task structure, thus, their scope is to the entire task. However, their accessibility, which defines whether or not the function is accessible and visible to an external task, can be defined in the following manners:

- **Public** - Visible and accessible to external tasks.
- **Private** - Not visible or accessible to external tasks. Access is limited to the containing type.

The ability to set task functions as **public** or **private** give developers greater control over how a task is used by another task.

Sub-Tasks

Sub-tasks are task files that are executed within another task by using the Task - Start sub-task activity. In this situation, the parent task's (that is, the task executing the Start sub-task step) basic functions, extended functions, public task functions, public task variables, and any local variables (not marked as private), created up to point where the Start sub-task step is encountered, are accessible to the sub-task. All other variables and functions are not accessible to the sub-task. Conversely, because the Start sub-task action is a synchronous operation, the sub-task's public functions and public task variables are not accessible from the parent task. This is fully backward compatible with previous versions of Automate Desktop, since in the past there was only one "function" (what is now called "main"), no task variables, and all (local) variables and extended functions were considered public.

When you define a task function, you can optionally define one or more named, typed values that the function receives as input (known as argument parameters). You can declare arguments as optional and set a default value for any argument - so if the argument is not present, the default value will be used. You can also define a type of value that the function will pass back to the task as output upon completion (known as return type).

Function argument parameters and return values are extremely flexible in Automate Desktop. You can define anything from a simple utility function with a single unnamed parameter to a complex function with expressive parameter names and different parameter options. You can provide default values to simplify function calls and can be passed as in-out parameters, which modify a passed variable once the function has completed its execution.

Creating and Defining Task Functions

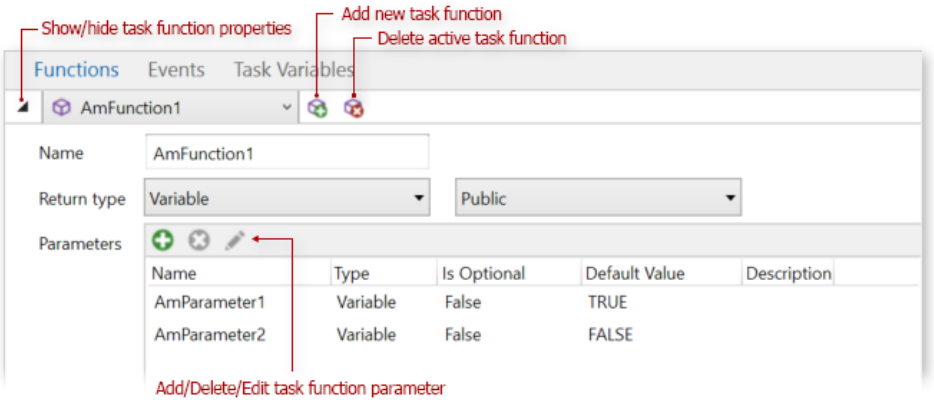
When you define a task function, you can optionally define one or more named, typed values that the function receives as input (known as argument parameters). You can declare arguments as optional and set a default value for any argument - so if the argument is not present, the default value will be used. You can also define a type of value

that the function will pass back to the task as output upon completion (known as return type).

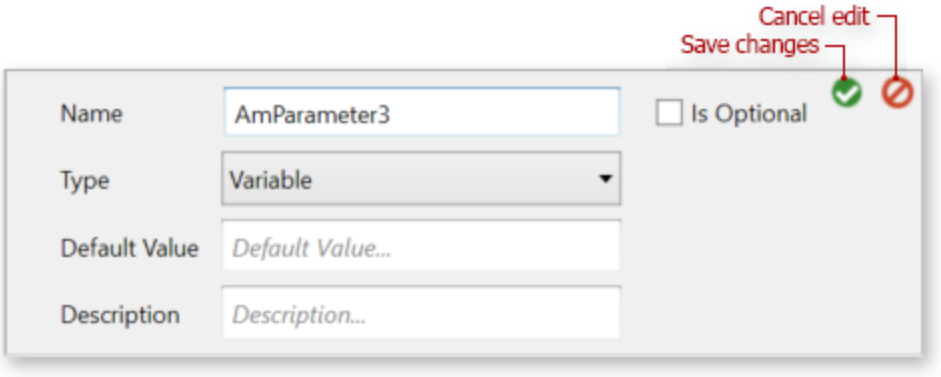
Function argument parameters and return values are extremely flexible in Automate Desktop. You can define anything from a simple utility function with a single unnamed parameter to a complex function with expressive parameter names and different parameter options. You can provide default values to simplify function calls and can be passed as in-out parameters, which modify a passed variable once the function has completed its execution.

To create and define a new task function

1. On the Task Builder's Steps panel, select Functions.
2. Click the **Add new function** button. This expands the UI panel to display available function properties.



3. Enter a function name, select a return type and specify whether the function is public or private (see the Function Properties table below for more details).
4. To add argument parameters, click the **Add new parameter** button. A dialog appears that allows you to enter function parameters.



5. Enter the required information and click the **Save Changes** button upon completion (see the Argument Parameters table below for more details). Perform steps 3-4 to add additional parameters. To edit or remove an existing parameter, select it from the list and click the **Edit** or **Remove** button.

To modify an existing task function

1. On the Task Builder's Steps panel, click the drop-down button to expand the list of existing task functions/events.
2. Select the task function you wish to modify, then click the (**View Function Properties**) button.
3. Perform the appropriate modifications.

To delete an existing task function

1. On the Task Builder's Steps panel, click the drop-down button that displays existing task functions/events.
2. Select the function you wish to delete.
3. Click the (**Remove Task Function**) button. This will permanently delete the task function.

Function Parameters

The following table describes the parameters available during task function creation/modification.

Property	Type	Description
Name	Text	The identifier by which the function can be called. This parameter defaults to the name *NewFunction* but can be modified to a name that better identifies the function.
Return type	Text (options)	The return value type. The available options are: • **Nothing** - No value (nothing) will be returned. • **Variable** - The return value will be in the form of a variable. • **Array** - The return value will be in the form of an array. • **Dataset** - The return value will be in the form of a dataset.
Function is private	Yes/No	If selected, the function becomes private.

Argument Parameters

The following table describes the parameters available during task function creation/modification.

Property	Type	Description
Name	Text	The identifier by which the function can be called. This parameter defaults to the name *NewFunction* but can be modified to a name that better identifies the function.

Property	Type	Description
Type	Text (options)	The return value type. The available options are: • **Nothing** - There will be no return value. • **Variable** - The return value will be in the form of a variable. • **Array** - The return value will be in the form of an array. • **Dataset** - The return value will be in the form of a dataset.
Is Optional	TRUE/FALSE	Specifies whether the parameter is optional (TRUE) or required (FALSE). Any call must provide arguments for all required parameters, but can omit arguments for optional parameters. Each optional parameter has a default value as part of its definition. If no argument is sent for that parameter, the default value is used. When you define a complicated function, you will often want to set this to optional (TRUE) for flexibility.
Default value	Text	The function's default value (if any).
Description	Text	An optional description to enter about the parameter.

Calling Task Functions

A function doesn't do anything until it is called. Every function has a function name which may describe the procedure that the function performs. To use a function, you “call” that function with its name and pass it input values (known as arguments) that match the types of the function’s parameters. If you are calling a function with no parameters, an empty pair of parentheses is required.

The Task - Call function activity calls a function defined in the task. If the function returns a result, you can specify the name of the variable in which to place the value. This activity is mainly used to pass control to a subroutine; after the subroutine is executed, control returns to the next instruction in main task.

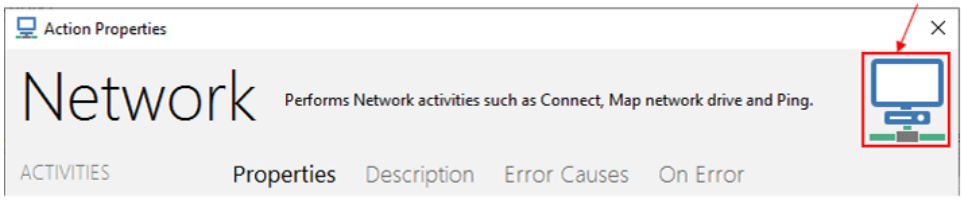
In addition, the Variable - Return activity can be used to exit the current function and return task execution to the calling function. A return value can also be specified to the caller. This activity immediately ends execution of the current function and returns its argument as the value of the function call.

Actions

The work items performed by a task are called actions or activities. They are the basic building blocks that enable development of intuitive tasks without the need to write code. Actions are organized in a hierarchical format in the Task Builder, each action containing individual activities (or sub-actions) which are subroutines directly related to the main action. You simply drag-and-drop actions / activities from the Task Builder Actions Panel onto the Task Builder Steps Panel to build a series of steps, which collectively make up the task.

> **NOTE:** The Help for the Actions is stored in **..\Program Files\Globalscape\AutoMate\Help**. Click **Default.htm** to open the help in your default browser.

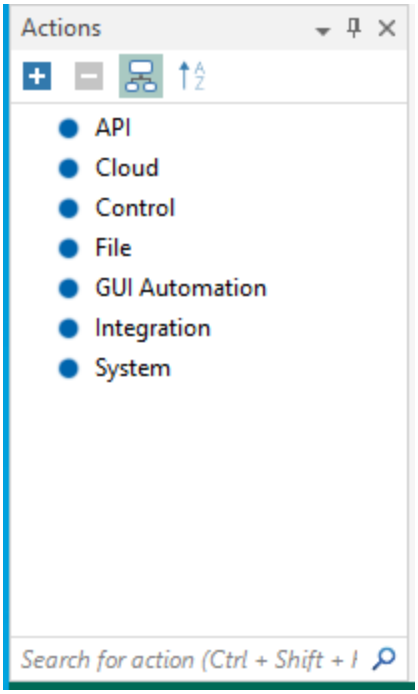
Actions and activities perform operations such as starting applications, sending keystrokes, clicking controls, uploading files, and much more. They can be further expanded by the use of variables , constants, and expressions. A task may comprise a single action or it may involve multiple actions and more complex elements such as variables, expressions, conditional logic, looping, and more.

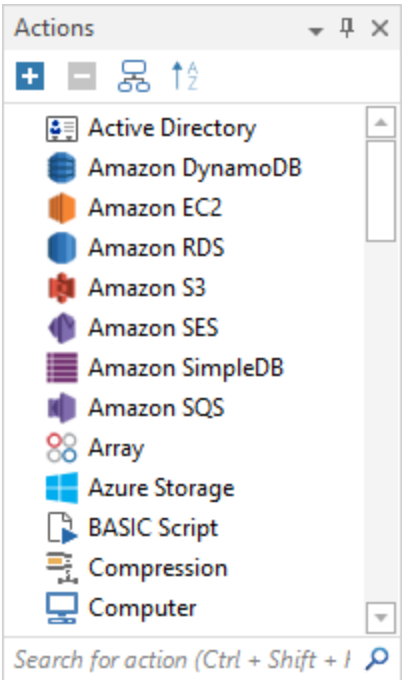
> **NOTE:** Refer to the Task Builder help content for details of the Advanced Workflow actions. You can also click the action's icon to go directly to the help for that action.
>
> 
>

Available Actions

The following table lists all of the actions available along with a brief description. To view more information about a specific action or to access the available activities (or sub-actions) for each action, open the action in the Task Builder and open the Help file.

In the Task Builder, the Actions are organized in the Actions panel by categories. You can expand the categories by clicking the organize icon to view the actions in alphabetical order. Or you can search for an action in the Search box at the bottom of the panel.





Name	Description
Active Directory	Automates a variety of common Active Directory tasks is available, such user account creation / provisioning and deprovisioning. Most Active Directory related jobs are often repetitive and time-consuming, so automating them does a world of good to network administrators and their enterprise, in terms of time saved and standards achieved.
Amazon DynamoDB	Allows developers to automate a variety of DynamoDB operations without the need to write code. Since DynamoDB is a service based on throughput, developers can also create tasks to monitor usage and storage capacity.
Amazon EC2 (Elastic Compute Cloud)	Allows developers to automate the management of EC2 instances, improving efficiency while maintaining a high level of performance. Using these activities, one can optimize the use of compute resources in a variety of circumstances.
Amazon RDS (Relational Database Service)	Provides cost-efficient and re-sizable capacity while automating time-consuming database administration tasks, freeing you up to focus on your applications and business.
Amazon S3 (Simple Storage Service)	Provides developers the tools to automate to automate common S3-related operations such as the creation and management of buckets and the objects that reside in them.

Name	Description
Amazon SES (Simple Email Service)	Enables automation of many common aspects of email management, such as creation, verification and retrieval of identities for a specific AWS account as well as composing and sending email messages based on input data.
Amazon SimpleDB (Simple Database)	Contains a variety of activities that enables automation of common to complex SimpleDB operations, eliminating the administrative burden of performing manual operations such as data modeling, index maintenance, and performance tuning.
Amazon SQS (Simple Queue Service)	Automates the creation of queues and movement of data between distributed components of relevant applications that perform different tasks without losing messages or requiring each component to be always available.
Array	Enables creation of arrays to be used during task execution, allowing a group of objects with the same attributes to be addressed individually. It can also set an array to a specific value or modify the current size of an array.
Azure Storage	Enables cloud storage automation with a comprehensive set of Windows Azure Storage activities, including blob, entity, table, queue and message- related activities.
BASIC Script	Often used to create sophisticated scripts without the need to write VBScripts, Shell Scripts, batch files or any type of code. In addition, it can also incorporate existing VBScripts, Powershell and other scripts into manageable task flows and add new functionality to old applications with the use of over 500 available actions and activities.
Compression	Automates the ability to compress and / or decompress data in a specific directory. It supports all major compression types, including ZIP, JAR, GZIP, TAR and 7ZIP (among others). Other features include the ability to password protect encrypted data and change the level at which files are to be compressed.
Computer	Allows the ability to automate various Windows start menu power button functions such as Lock, Restart, Shut down, Log off, Suspend, and Hibernate.
Cryptography	Automate Desktop's encryption and decryption of files. Can also create or delete key containers, generating both public and private key files from existing key containers and digitally signing files using newly generated private keys. Supports PGP (Pretty Good Privacy), one of the most widely used cryptography systems.
Database	Enables creation of sophisticated database-driven applications without the need to write code. Supports SQL Server, Oracle, Access, Sybase, DB2 and other ODBC-based database. This powerful tool can automate report generation and distribution, run DTS or similar database maintenance processes automatically and expand tasks to do even more using Automate Desktop's library of actions and activities.
Dataset	Automates the creation or cloning of a dataset, as well as managing the rows and columns within it.

Name	Description
Dialog	Contains various dialog activities, each displaying its own unique dialog window during runtime. The type of dialog displayed is dependent upon the desired user interaction. Additionally, a Custom Dialog activity can be used to create your own dialogs in situations where a normal Windows message box may not be sufficient enough.
DLL (Dynamic-link Library)	Executes the specified method in a managed or unmanaged dynamic-link library (DLL) and stores the return value of the method in an Automate Desktop variable.
Dynamics CRM	Automates a variety of common CRM operations in order to ensure consistency in how businesses interact with their customers. It can be used for creating, updating or deleting contacts, retrieving audit records, tracking specific information, and uploading and downloading attachments.
Email	Combined with the Exchange action, Automate Desktop presents an effective email attendant that allows you to automate common to complex email operations and simultaneously manage several email accounts from as many mail servers as required.
Environment Variables	Automates the creation, manipulation and management of environment variables in a Microsoft Windows-based operating system without the use of scripts or the command line. Activities include the ability to get, set, list and delete environment variables as well as append data to an existing environment variable.
Event Log	Allows custom events to be logged in a purely automated fashion. Developers can control the type of event to log by specifying a level value such as warning or error. Furthermore, this action can be used along with the Event Log trigger as a way to launch other tasks upon arrival of a specific event.
Excel	When you work with Microsoft Excel, you usually select one or more cells and then perform an action, such as formatting the cells, entering values in them or retrieving current values. The Excel action encompasses a variety of activities that are capable of automating such operations, thus, saving time and eliminating errors. In addition, this action is capable of running existing Excel macros as part of a fully automated sequence.
Exchange	Comprises a variety of activities that can be used to automate common operations on Microsoft Exchange objects, such as the ability to create calendars or appointments, delete contacts, modify tasks and retrieve email. This action supports Exchange 2003 which uses the WebDav protocol as well as Exchange 2007 and 2010, which uses the EWS (Exchange Web Service) protocol.
File System	Encompasses a comprehensive collection of activities for automating all file and folder-related operations. Activities include the ability to copy, delete or rename files and folders, read from a file and write to a file. Other activities include the ability to retrieve information about files, such as metadata and checksum values as well as the ability to split or concatenate files and synchronize folders (among others).

Name	Description
FTP (File Transfer Protocol)	Used to automate any FTP operation including downloads, uploads, renames, folder creations, deletions and more. Automate Desktop's internal FTP engine is compatible with all FTP connection methods, particularly security-based connections. This includes both explicit and implicit FTPS (FTP Secure or FTP SSL) that adds support for the Transport Layer Security (TLS) and the Secure Sockets Layer (SSL) cryptographic protocols.
HTTP (Hypertext Transfer Protocol)	Contains activities that are ideal for automating common HTTP operations, such as GET, PUT, POST, DELETE and more. HTTP requests are sent directly instead of automating a browser to perform such operations. This action also supports Hypertext Transfer Protocol Secure (HTTPS) protocols, which is a combination of HTTP with the SSL/TLS protocol to provide encrypted communication and secure identification of a network web server.
If	In programming, the **If** statement is used to run code based on conditions. This allows programs the ability to perform different operations in response to different conditions. Automate Desktop's If activities encompass the same capabilities but with a more user-friendly approach. They enable complex decision making based on variable data in the context of a task.
Image	Developers and designers must deal with editing and converting many images by hand, which can be time consuming. Automate Desktop's Image action consists of activities that can greatly reduce the time it takes to perform such a process. For example, you can create a single task that reduces the size of an image, applies a filter for a particular effect and converts the image file to the desired format. In addition, you can integrate loop related activities to run this task on hundreds or thousands of images.
JSON	Contains activities that allow you to decode and encode JSON (JavaScript Object Notation) objects
Label	Labels are used to label a step in a task with a specific name. A label is used with a Goto activity and/or the On Error Goto option (found in the On Error properties) to allow redirection of task execution to the label specified.
Loop	In computer programming, a loop is a fundamental yet powerful technique that is commonly used in writing programs. A loop is a sequence or "block" of instructions that is continually repeated a specified number of times or until a certain condition is reached. It allows a very simple operation to produce a significantly greater result simply by repetition. Similar to the concept of loops in programming, the Loop action in Automate Desktop allows repeated execution of a group of steps. Activities include the ability to loop through datasets, files, processes, lists, and more for dynamic automation.

Name	Description
MSMQ (Microsoft Message Queuing)	MSMQ (Microsoft Message Queuing) technology enables applications to communicate with each other in an effective and dependable manner, even in unreliable distributed environments where intermediate servers or systems may not always be available. This is made possible due to the fact that messages are not exchanged directly between applications but rather through a message queue, which is a temporary storage location from which messages can be sent and received reliably. Applications that run at different times can essentially send messages to queues and receive messages from queues at their convenience. Automate Desktop's MSMQ action provides a reliable and secure way to automate message delivery and retrieval by way of MSMQ. It contains individual activities that lets you to create, send, retrieve, clear, delete, or wait for MSMQ messages on a local or remote machine.
Network	Contains activities used to automate various network related operations, such as establishing a network connection, mapping, or unmapping a network drive or sending a "ping" request to a computer.
OCR (Optical Character Recognition)	Provides new opportunities and techniques with regards to combining OCR capabilities in an automated environment. It enables you to convert typewritten, handwritten or printed text as well as text contained in images to an Automate Desktop variable or dataset, making it possible to search for a word or phrase, store text more compactly, and apply techniques such as text mining or text to speech.
OpenDocument Spreadsheet (ODS)	The OpenDocument Spreadsheet action is similar to the Excel action. Unlike the Excel action, however, the OpenDocument Spreadsheet action enables creation and management of spreadsheets without requiring Microsoft Excel to be installed on the system. This action not only supports spreadsheet generation and other basic file formatting features, but also supports a number of advanced features such as importing as well as exporting spreadsheet data, reading, writing, and modifying XLSX, XLS, ODS, CSV, or HTML files. This makes it much more convenient for developers to manipulate spreadsheet contents, format cells, and protect files.
PDF (Portable Document Format)	Automates an assortment of common PDF operations. Activities include the ability to concatenate or merge two or more PDF files, append new pages to an existing PDF file, extract pages, insert pages at a particular location in an existing PDF file or split one PDF into two or more PDF files (to name a few). For added Security, this action provides the ability to encrypt and decrypt PDF files with either 40-bit,128-bit, or 256-bit encryption and also specify user passwords for further protection.

Name	Description
PowerShell	PowerShell is a command line shell and scripting language for Microsoft Windows that is built on the .NET framework. PowerShell's task automation framework allows administrators to perform administrative tasks on both local and remote Windows systems. Automate Desktop enables management and control for PowerShell processing. With the use of the PowerShell action, you can embed existing PowerShell scripts to a task or point to an external .PS1 file to allow for seamless integration with other actions. To run unattended or scheduled PowerShell operations, simply add a schedule trigger.
Printer	Can optimize printing resources and print documents when they are required, not necessarily when they are generated. This action includes activities that can automate print jobs across various print queues, change the default printer or remove a specific printer on the fly. It also contains an activity that can retrieve status sand other properties about any printer across the network so you can better manage which printers and at what time print jobs should take place.
Processes	Enables automation of most process management operations. This action comprises activities that can automate the act of starting, stopping or controlling a process as well as query a process for specific types of information, such as names of loaded modules, number of consumed handles and sets of threads, start time and much more. You can also adjust the priorities of running programs so that badly behaved processes won't negatively impact the responsiveness of your computer. The available activities for this action supports management of local or remote processes.
Registry	Encompasses a variety of activities capable of automating many common registry configuration and management operations, such as the ability to create, manipulate, rename, and delete registry keys, sub-keys or value data. This action also allows registry values to be read and stored, which is ideal in situations when you may need to retrieve configuration data, for instance, to keep track of the software you install on the computer or save data collected from an external application so that it can be retrieved and used the next time the task is run.
Run	Runs the program, application or document specified. If a document is specified, the application associated with the document will be used to open it. This action comprises some built in conveniences, such as the ability to wait for an application to finish loading (that is, be ready for input) or wait for it to complete its operation before continuing to the next step. Commonly used to start an application or open a document in preparation to send keystrokes, mouse clicks or other interactive steps.
Security	Enhances the security of an automation routine by generating a strong password and / or suspending task execution until the user properly authenticates himself / herself by either entering the correct password. Can also be used to suspend execution until a specific a specific user is logged on.

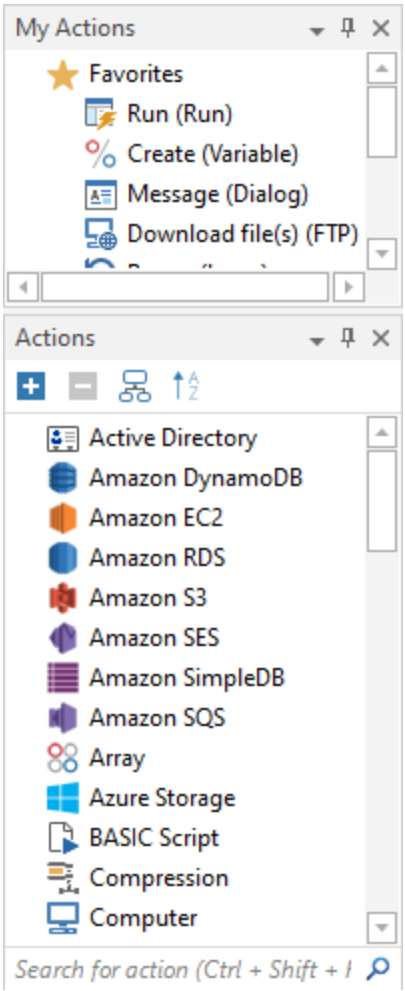
Name	Description
Select	A useful alternative to using an "If" statement in a task. Like any If activity, this action allows decisions to be made during task execution depending on the result of an expression. The difference is the speed and readability of the resulting task when one of several decisions can be reached. For example, for a simple expression such as 1=1 (true) or 1>2 (false), it would be best to use the If activity. For more complex decisions where there are several possible answers such as favorite color=red, blue, green or yellow, Select / Case activities would be a better choice. This is because Select / Case presents the task steps in a manner that is more readable, and at runtime, the evaluation of the expression occurs only once rather than needing repeated If activities with individual comparison.
Services	Provides various ways to automate the management of local or remote services. Using a variety of activities, you can easily automate operations such as starting, stopping, installing, or uninstalling services. You can also query for a list of services installed on the system along with their current state.
SharePoint	Comprised of over 40 activities that enables automation of simple to complex SharePoint operations. For instance, this action can automate the creation, management and implementation of SharePoint sites that are discoverable throughout the organization. By further streamlining and automating SharePoint processes, a company's resources can be allocated to other areas, which can lead to reduced costs and greater operational efficiency.
SNMP (Simple Network Management Protocol)	Used to automate common SNMP commands such as GET, GET BULK, GET NEXT, SEND TRAP, SET, and WALK, providing server administrators and IT personnel a means to monitor a complete network infrastructure through simple Automate Desktop tasks. Also enables integration of Automate Desktop with products such as Microsoft MOM and others.
Task	Allows task level operations to be automated during runtime. Activities include the ability to start a secondary task (also known as a child task or subtask) from a parent task or start another task as a separate thread. Other activities can disable / re-enable execution of other tasks. For instance, a Disable task activity could be placed at the beginning of Task1 to disable execution of Task2 for the time being to ensure it doesn't interfere with execution of Task1. An Enable task step could be placed at the end of Task1 to re-enable execution of Task2.
Terminal	Often used to completely simulate the procedures that a user typically performs during a terminal session. The Terminal action allows legacy systems to be automated in the background without the need for 3rd party software. With Automate Desktop, terminal servers can be used to run programs, save files, and employ network resources, altogether, working as a bridge to connect terminal, and / or legacy systems to Microsoft Windows networks and computers.

Name	Description
Text	Text manipulation is a way to administer and modify textual data, including text conversion, transformation, and extraction. Some common text operations involve stripping of characters or white space, converting words to upper or lower case, adding or replacing characters, and calculating character count for the selected text. Automate Desktop encompasses a variety of activities primarily designed to automate the management, modification, and enhancement of text. These time saving activities are the complete solution for any data and text editing procedures that are performed in large quantities or in a repetitive manner.
Timer	A timer can be used to examine the sequence of an event or process taking place. This is an ideal way for developers to gauge the total execution time of an individual task step or series of steps that perform time sensitive operations. A Start Timer activity creates a new session and begins the recording process and a Stop Timer activity marks its ending. The elapsed time is then output to an Automate Desktop variable. In addition, a Read Timer activity can be used to gauge the execution time between Start Timer and Stop Timer steps. Multiple timer operations can occur simultaneously within the same task as long as each operation is designated by a unique session.
Type	Used to create an Automate Desktop object from the defined custom type or web service. This action can also be used to define a custom object type or object types defined in a Web Service or import Web Service or .Net assembly types.
Variable	The Variable action comprises various activities geared for the creation and management of variables and / or arrays. The developer can enter a value to initially populate the variable during creation or the value field can be left blank. Instead, variables can be set with a value using the Set Variable activity, which adds or changes the contents of an already existing variable. Certain available actions that support populating variables can also set or modify a variable's contents, such as the Input Dialog activity, which displays an input box allowing the user to enter a value which is saved to the variable specified.
VMWare Guest	Useful for automating many common operations on a VMware guest operating system, such as running a program or script, managing processes, creating, and managing files and directories, creating screen shots, administering environment variables, and much more. With these activities, you can run common tasks on a network where you have multiple virtual machines running on one or more physical machines. Automate Desktop supports the full suite of virtualization software including ESX, ESXi, Workstation, and Player.

Name	Description
VMWare Host	Useful for automating many common performance and auditing operations on a VMware host operating system. For instance, you can create a task to automate the process of modifying a limitless number of hosts, freeing up time for more strategic activities. Automate Desktop can automatically generate lists of running hosts as frequently as required as well as automatically register and unregister hosts based on any event defined. Other activities include the ability to start, resume, or suspend a VM, take snapshots, revert to a previous snapshot, and much more.
Wait	In programming, a wait state is a situation in which a program waits for the occurrence or completion of a specific event or condition before continuing. Because the result of a wait can largely impact task execution, many Automate Desktop activities have their own built-in wait capabilities. In addition to the built-in wait function, a number of individual 'Wait' activities are also available which can be used at any point inside a task without the need to write code. These activities provide additional intelligence to Automate Desktop's already powerful task flow capabilities by allowing execution to pause until a specific date / time, until the existence (or non-existence) of a file or window, wait for a process to start or end, or wait for a specific amount of time.
Web Service	The benefits of Web Services to any business can be summed up in one word: efficiency. Automate Desktop helps leverage this functionality with its Web Service - Execute function activity. This action allows you to automate the ability to execute Web Service calls by way of WSDL (Web Service Definition Language) URI (Uniform Resource Identifier) path. With Automate Desktop, once you enter a WSDL URI path and select **Go**, it automatically evaluates that Web Service and fills in the information for you to select from a list.
WMI (Windows Management Instrumentation)	Can be useful for monitoring and controlling managed resources on a local or remote computer, thus, improving manageability of computers in a networked environment. For instance, a WQL Query task can initially determine any issues and direct itself to perform proper activities to correct the problem before proceeding to other steps.
XML (Extensible Markup Language)	Automate Desktop's XML action enables automation of many common XML related activities, such as transforming, merging, validating, and signing XML documents. They are also capable of reading, creating, editing, or deleting XML nodes as well as extracting XML fragments (that is, a collection of XML nodes) and saving them to a file or other means. The XML activities provide all of the features and functionality necessary for managing and manipulating XML.

Actions Panel

A task is something that performs a number of sequential steps that are necessary to reach a desired outcome. The Actions panel houses all of the available actions and associated activities that make up the steps to be performed during task execution. Actions and activities are assembled visually by dragging them from the Actions panel onto the Steps panel (or by simply double-clicking the desired action/activity) and entering their properties. When this is accomplished, the action becomes a task step. This panel also contains a My Actions view which puts your favorite, recently used and most used activities along with any saved snippets in a convenient location for easy access.



Actions view

The Actions view is comprised of over 400 built-in actions and activities, which range from starting an application, opening a web page, focusing on a window, uploading or downloading files, sending and receiving email, simulating mouse or keyboard activity and much more. Actions and activities are arranged in a hierarchical format so that users can browse through them conveniently. To view a list of activities (or sub-actions), expand the main action by double-clicking its associated icon or by clicking the small arrow to the left of the action. For more details, see Finding & Organizing Actions.

Once the desired activity is found, simply drag it onto the Steps panel or double-click it to make it a task step. In addition, the action itself can be dragged into the Steps panel. Doing so will display the properties dialog where an activity can be selected from a list. For more details, see Setting Action Properties.

> **NOTE:** Unlicensed actions and activities will appear opaque. Double-clicking, dragging or adding them to the Steps panel in any other way will generate an error.

Search dialog

The bottom portion of the Actions panel contains an intuitive search dialog that enables you to define search criteria and provide more efficient and effective searches. It supports dynamic search filtering, allowing you to enter all or part of an action's name and view only actions that contain matching text.

To find an action/activity

1. Click inside the Search dialog or press CTRL+SHIFT+F on your keyboard.
2. Enter all or part of an action's name. The list is updated as you type each letter. The list is updated as you type each letter.
3. To empty the Search dialog of its contents, do one of the following:
    - Click the "X" button located on the right side of the Search dialog (this button only appears when the Search dialog is populated with characters).
    - Press ESCAPE on your keyboard.

Context menu

By default, the Actions view displays all available actions in ascending alphabetical order and associated activities (or sub-actions) are hidden from view. Alternately, all actions can be viewed in descending alphabetical order and expanded to display all activities individually. All options that assist in locating and administering Automate Desktop actions as well as customizing the Actions panel are contained in a context menu that you can open by right-clicking a specific action or an empty portion inside the window Actions panel.

Item	Description
Add Step	Adds the selected action/activity to the Steps panel and automatically places it at the end of the task (for example, last step). This option is available only when a right-click is performed on a specific action/activity.
Insert Step	Inserts the selected action/activity directly above the currently highlighted step in the Steps panel. This option is available only when a right-click is performed on a specific action/activity.
Sort A to Z	Sorts the individual action folders in ascending or descending alphabetical order.
Expand All	Expands all categorized folders. This option is available only if the **Categories** option is enabled.
Collapse All	Collapses all categorized folders. This option is available only if the **Categories** option is enabled.
Add to Favorites	Adds the selected action to your **Favorites** list. This option is available only when a right-click is performed on a specific action.
Help (F1)	Opens the selected action's help topic. This option is available when a right-click is performed on a specific action. Selecting an action and pressing F1 also opens that action's help topic.

My actions view

Automate Desktop's array of structured actions and activities can automate virtually any type of business process one can think of. Nonetheless, most developers find themselves

utilizing just a handful of activities specific to the needs of their business or department. The My Actions view allows easy access to such activities by way of Jump Lists. That way, you don't have to scroll through numerous actions or type a keyword in the Search bar to find an activity. Simply select the activity from the appropriate list in the My Actions panel. For additional details, see My Actions.

Favorites

The **Favorites** folder can be used as a placeholder for a collection of preferred actions and activities. Favorites can be created and managed by the user themselves. When an activity is defined as a favorite, a copy of that activity is added to the Favorites list. That way, users won't need to spend time searching through the long list of available actions. They can simply select the activity from their Favorites list. Users can also reorder their favorites by title or create folders/sub-folders to consolidate common favorites into specific categories, forming a more organized development environment. Actions and activities can be added to your favorites by way of the Actions panel or Steps panel. Once added, any favorites can be incorporated into task steps by way of drag and drop or by a double-click action.

To add an action or activity from the Actions Panel to Favorites

- Drag the desired action from the Actions panel onto the Favorites folder or any sub-folder.

  OR

- Right-click the desired action and select **Add to Favorites** from the context menu.

To add steps from the Steps Panel to Favorites

- Drag the desired steps from the Steps panel onto the Favorites folder or any sub-folder. To select more than one step, hold down CTRL during selection.

  OR

- Right-click the desired steps and select **Add to Favorites** from the context menu.

Snippets

For those who use specific activities that contain commonly used properties or settings, such activities can be saved in the **Snippets** folder. Snippets provide an easy way to implement frequently used code into a task. Instead of re-entering the properties for a specific activity or series of activities every time they're added as task steps, users can save existing steps as a snippet and simply drag and drop the snippet wherever it is needed in any task. A snippet can consist of one or more steps. Once a snippet is added to a task, its contents act as normal steps (that is, they can be reorganized or deleted or their properties can be modified as needed). Users can reorder their snippets list by title or create folders/sub-folders to consolidate common snippets, thus, creating a cleaner development environment.

To add one or more steps from the Steps Panel to Snippets

1. Drag the desired steps from the Steps panel onto the Snippets folder or any sub-folder or simply right-click the desired steps and select **Add to Snippets** from the context menu. To select more than one step, hold down CTRL during selection.
2. Enter a unique name for the new snippet.

> **NOTE:** Users will be able to save an unlicensed activity as a snippet and drag and drop that snippet into the Steps panel. However, once in the task panel, it should continue to behave as an unlicensed activity.

The **Recent** folder is populated with the most recently used actions and activities. The **Frequent** folder is populated with the most frequently used actions and activities. Similar to **Snippets** and **Favorites**, these folders are designed to make it easier and more convenient for you to find what you want by placing the actions and activities that you mostly use in a convenient location. By default, 10 items are displayed in the **Recent** and **Frequent** folders.

Context menu

Right-clicking an action located inside the **My Actions** panel will display a context menu with the following options.

Item	Description
Create Folder	Creates a folder at the current position. Allows you to better manage the actions and activities you use the most. This option is available only if a right-click is performed on an activity from your Favorites or Snippets list.
Remove	Removes the selected activity from its current position. This option is available only if a right-click is performed on an activity from your Favorites or Snippets list.
Reorder by Title	Reorders existing sub-items by title. Active only if a right-click is performed on a parent item.
Clear	Removes the selected activity or snippet from the My Actions panel.
Clear All My Actions	Clears the My Actions panel of all activities and snippets

Finding & Organizing Actions/ Activities

There are over 500 available actions and activities that you can use to create the steps of your task. This may seem overwhelming to some individuals, especially new or novice users. Therefore, the Task Builder provides a variety of ways to assist in easily finding actions or activities you are searching for. In addition, it keeps track of a user's favorite, most often and most recently used actions/ activities and places them in the My Actions panel for quick and easy access.

Finding actions & activities

The bottom portion of the Actions panel contains an intuitive search dialog (highlighted in red below) that enables you to define search criteria and provide more efficient and effective searches. It supports dynamic search filtering, allowing you to enter all or part of an action's name and view only actions that contain matching text. The list is updated as you type each letter.

To search for an action or activity

1. Click inside the search dialog or press CTRL+ SHIFT + F to automatically place the cursor inside the dialog.
2. Type all or part of the action name you want to search for. The list of relevant actions narrow as you type each letter.
3. Use the UP or DOWN arrows to navigate to the desired action then press ENTER or click and drag the action to the Steps pane.
4. To clear the search, click "X" located on the right side of the search dialog or press ESC.

Organizing actions & activities

By default, actions are categorized in ascending alphabetical order and activities (or sub-actions) are categorized in a logical sequence. Alternately actions can be viewed in descending alphabetical order or they can be expanded to reveal all activities. Most commands and operations relevant to actions are contained in the **Actions** tab of the ribbon (as shown below).

To change the sort order of actions

1. Right-click anywhere in the Actions panel and select the context menu item labeled **Sort A to Z**. The order is changed to ascending or descending alphabetical order (depending on the original sort order).
2. Repeat step 1 again to switch back to the previous sort order.

> **NOTE:** Sorting only applies to actions. They don't apply to activities which are ordered in a fixed logical sequence.

To expand or collapse actions

1. To expand the actions list, revealing the list of activities for each action (if any), do one of the following:

- On the ribbon, select the **Actions** tab and click the **Expand All** button.
- Right-click anywhere in the Actions panel and select the context menu option **Expand All**.

2. To collapse the actions list, do one of the following:
   - On the ribbon, select the **Actions** tab and click the **Collapse All** button.
   - Right-click anywhere in the Actions panel and select the context menu option **Collapse All**.

My Actions

Having quick access to common items is essential for a productive work environment. Task Builder supports this capability with the My Actions pane. Similar to Jump Lists in Windows that enable quick access to frequently and recently used items, My Actions can take you right to the actions and activities you turn to each day. **The My Actions pane is designed** to make it easier for you to access common actions or recently used activities quicker by conveniently placing them in common folders directly above the Actions view (circled below).

You can use My Actions to open actions/activities, and you can also "pin" favorites, so you can quickly get to the actions that you use every day. You can drag an action from one of the folders in My Actions directly to the Steps panel. In addition, you can save actions that contain common properties as "snippets." Snippets store the current properties set for the action so yo can reuse it in different tasks.

The My Actions view houses four types of items:

- Favorites
- Snippets
- Recent/frequent

Favorites

Favorites can be used as a placeholder for a collection of preferred actions and activities. When an activity is set as a favorite, a copy of that activity is added to the **Favorites** folder

located in the Actions panel (highlighted in red below). That way, users spend less time searching through the long list of available actions and activities. They can simply select the activity from their Favorites list. Individual folders can be created to consolidate common activities into specific categories, forming a more organized development environment.

> **NOTE:** You can add actions and individual activities to your favorites list. By default, the list is set in chronological order, starting with the earliest addition and following the order in which they occurred. However, you can resort the list in alphabetical order by right-clicking **Favorites** and selecting **Reorder by title**.

To set an action or activity as a favorite

1. In the Actions panel, navigate to the action or activity you wish to add to your favorites list (only one action/activity folder can be selected at a time).
2. Right-click the action/activity and select **Add to Favorites** from the context menu that appears.

In addition, you can select the steps that perform the activity you wish to add to your favorites in the Steps panel. To select more than one step, hold down CTRL during selection, and then right-click the steps and select **Add to Favorites** from the context menu that appears.

To add a folder/sub-folder to Favorites

1. In the Actions panel, right-click **Favorites** and select **Create Folder**. A new folder is added to your favorites list.
2. Rename the newly created folder to the desired name.
3. Add sub-folders under the newly created folder by right-clicking that folder, and then selecting **Create Folder**.
4. Add actions/activities to a folder under **Favorites** by clicking and dragging them into the folder of your choice.

To remove actions, activities, or folders from Favorites

1. In the Actions panel, under **Favorites**, select the action, activity, or folder you wish to remove (only one action/activity/folder can be selected at a time).

2. Right-click the action/activity and select **Remove**.
3. To clear **Favorites** of all items (including actions, activities and folders), right-click on **Favorites** and then select **Clear** from the context menu that appears.

Snippets

For those who use common activities that perform identical operations in many tasks, such activities can be saved as snippets. Snippets provide an easy way to implement frequently used steps into a task. Instead of re-entering the properties for a specific activity or series of activities every time they're added as task steps, you can save the steps as a snippet and simply drag and drop the snippet wherever it is needed in any task. The main idea is to make the process of reusing as easy as possible to avoid wasting your valuable time to enter the same properties again.

A snippet can consist of one or more task steps that perform common actions. For example, a developer may create a basic Input - Send keystrokes snippet to enter specific keys when a user opens a particular window or clicks a specific button. Other snippets might be used to perform Dialog - Open file and Dialog - Save file operations.

Once a snippet is added to a task, its contents act as normal steps (that is, they can be reorganized or deleted or their properties can be modified as needed). Task steps can be easily saved as a snippet by way of drag-and-drop. Sub-folders can be added to the Snippets folder so developers can easily organize common or related snippets into categories, creating a cleaner, more manageable development environment.

> **NOTE:** Users will be able to save an unlicensed activity as a snippet and drag and drop that snippet into the Steps panel. However, during Runtime, the unlicensed step will fail, generating an "unlicensed" error.

To add one or more steps to snippets

1. Highlight the desired steps from the **Steps** panel. To select more than one step, hold down **CTRL** during selection.

   > **NOTE:** Selected steps do not need to be sequential.

2. Right-click on the selected steps, and then select **Add to Snippets**.

To add a folder/sub-folder to snippets

1. Right-click on **Snippets**, and then select **Create Folder**. A new folder is added to your favorites list.
2. Rename the newly created folder to the desired name.
3. Add sub-folders under the newly created folder by right-clicking that folder, and then selecting **Create Folder**.
4. Rename the newly created sub-folder to the desired name.

Recent and Frequent Lists

Task Builder automatically saves a history of recently and frequently used actions and activities, and displays them in Jump Lists under the **Recent** and **Frequent** folders (shown below) so you can easily "jump" to these items with a click of a button. The lists are automatically populated based on how frequently and how recently items have been used. For example, if the last activity you opened or used in a task was a VMWare Guest - Delete file activity, the next time you open Task Builder, this activity will be contained in your **Recent** folder, and possibly, your **Frequent** folder as well, depending on how often you used the activity in the past. By default, the **Recent** and **Frequent** folders are populated with up to 10 entries.

Having access to frequent and recent lists is great if you want quick access to often used and popular actions and activities without having to browse in the Actions list or perform a search, but not great if you're a privacy freak who doesn't want someone random to see what kind of actions/activities you often use if, for example, you use a shared or public computer, or you may just want to clear this history periodically as a security measure. Task Builder makes it easy to clear your recent and/or popular list of actions/activities or clear all Jump List items altogether.

> **NOTE:** Clearing **Jump List** items removes items from your **Recent** and **Frequent** lists. It doesn't delete the items from your computer permanently.

Action Properties

A major point of task construction is entering the properties required for each of the actions to be included as task steps. Accessing an action's properties can be can be accomplished within the Task Builder in a number of ways:

- Dragging an action from the Actions panel and dropping it into the Steps panel.
- Double-clicking an action from the Actions panel or right-clicking the action and selecting **Add step** or **Insert step** from the context menu.
- Double-clicking an existing step from the Steps panel or right-clicking the step and selecting **Properties** from the context menu.

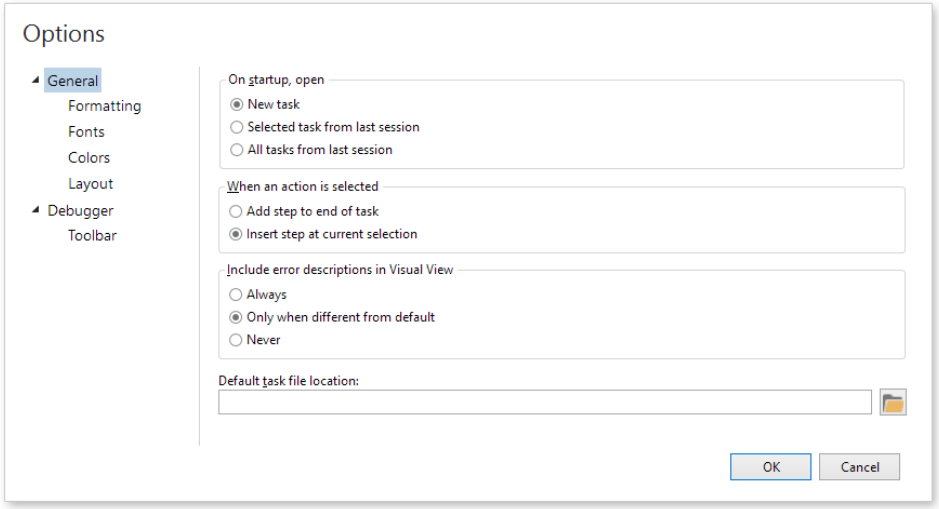
A dialog window appears titled Action Properties (also known as the Action Editor) displaying the selected activity's properties categorized into separate tabs. The left portion of the editor contains the full list of consolidated activities common to a given action. This enables jumping from one activity to another without the need to re-enter required parameters or re-open the action editor.

Depending on the action being edited, certain parameters may need to be entered or validated in order to properly add the action to the Steps panel. After the properties are set and the proper parameters assigned, clicking the **OK** button saves the settings and closes the properties dialog. The action then becomes a step and its settings are then displayed in the Steps panel.

> **NOTE:** Information regarding the properties and parameters available for any given action can be accessed in the Task Builder by clicking the **Action icon** or pressing F1.

Task Builder Options

The Task Builder contains its own set of options which can be configured to meet a user's specific necessities. They include color, icon and font settings, viewing options, debug preferences and formatting properties. Task Builder options can be accessed by way of **File > Options**.



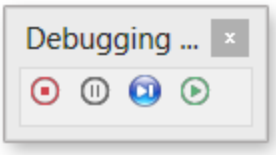
Available Options

Task Builder options are grouped into six separate categories, each encompassing specific preferences and options. Below lists the available categories, with links to the topics covering each.

Option	Description
Color Options	Contains options that determine the color to be used for the Task Builder interface or allows adjustments to be made to the current color scheme.
Debugger Options	Contains options that affect Task Builder's behavior while running and debugging tasks.
Fonts Options	Contains options to modify the font name, size and style to be used for the Visual view and the AML view.
Formatting Options	Contains options that deal with how steps should be formatted in the Steps pane.
General Options	Contains options that determine certain behavioral characteristics of Task Builder. Also determines the default managed task location.
Layout Options	Contains an option to reset Task Builder back to its original (default) layout.
Toolbar Options	Contains options that determine the look and feel of Automate's debugger toolbar.

Task Builder Toolbar Options

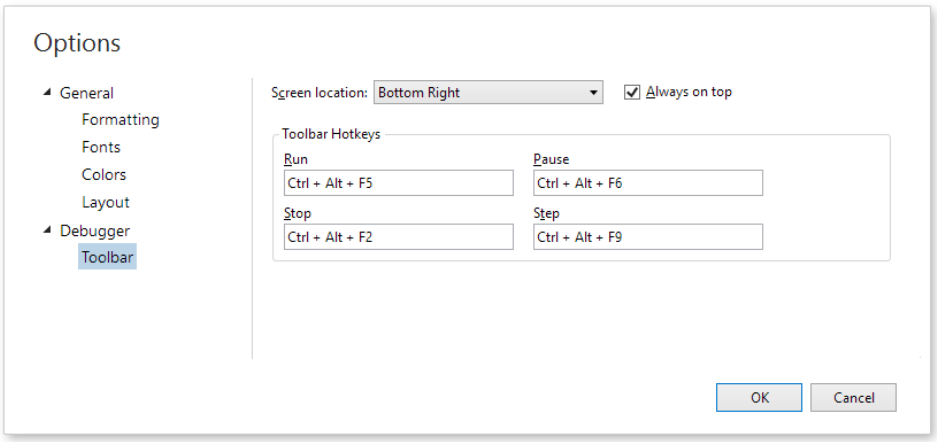
The Debug toolbar (illustrated below) is a small floating window that appears when a task is run from the Task Builder. It contains buttons to stop, pause, re-run and step through an active task to further aid you during the development stage. The Debug toolbar automatically closes upon completion of a running task or you can close it at any time by clicking "x".



The Debug toolbar supports hot-keys (also known as shortcut keys) that can be used as an alternate control mechanism. You can change the default toolbar shortcut keys to any desired key combinations not already in use, and you can change the position of the Debug

toolbar on the screen. In addition, you can choose to show or hide the Debug toolbar or have it always appear in the foreground or background.

To access the Toolbar Options page in the Task Builder, select **File** > **Options** > **Debugger** > **Toolbar**. Toolbar options is a subset of Task Builder Options.



Parameters

The following table describes available Toolbar options:

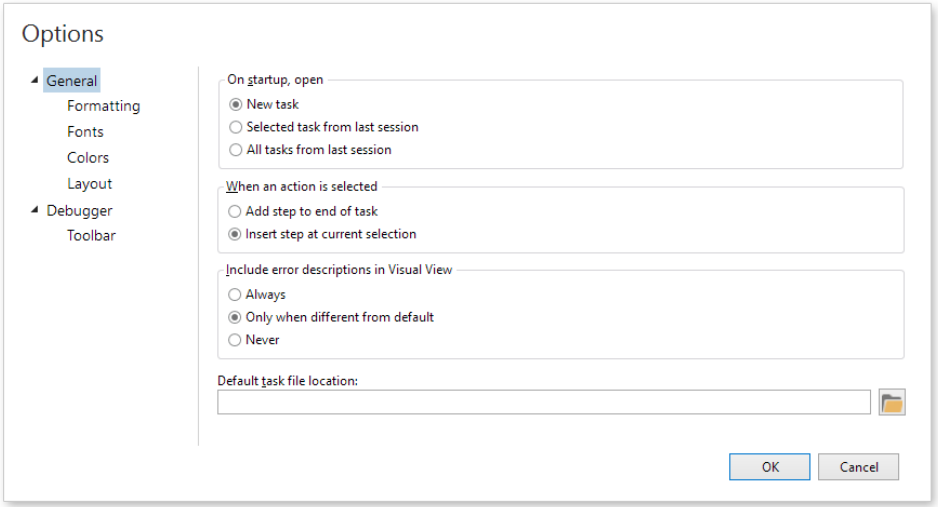
Property	Default	Description
Screen Location	Bottom Right	Specifies the area on the screen that the debug toolbar should be displayed. The available options are: • **Do not show** -The debug toolbar will not appear during task execution. • **Top Left** - The debug toolbar will be displayed on the upper left corner of the screen. • **Top Right** - The debug toolbar will be displayed on the upper right corner of the screen. • **Bottom Left:** - The debug toolbar will be displayed on the lower left corner of screen. • **Bottom Right** - The debug toolbar will be displayed on the lower right corner of screen.
Always on Top	Enabled	If enabled, the toolbar will always be in the foreground, in front of other windows. If disabled, the toolbar will appear in the background. This parameter is enabled by default.

Property	Default	Description
Toolbar Hot-keys	• Run: CTRL+ALT+F5 • Stop: CTRL+ALT+F2 • Pause: CTRL+ALT+F9 • Step: CTRL+ALT+F6	Allows entry of specific shortcut keys (also known as hot-keys) used to run, stop, pause or step through a task. The available options along with their default hot-key parameters are as follows: • **Run** - Runs the entire task, from beginning to end, or until an error is encountered. • **Stop** - CTRL + ALT + F2 • **Pause** - CTRL + ALT + F9 • **Step** - Runs the task step by step, pausing on each step. **NOTE:** For more details about using the various run options in Task Builder as a debug tool, see Run Options.

Task Builder General Options

Task Builder contains general options that enables control over a variety of interactive aspects. You can choose whether double-clicking an action/activity from the Actions panel should add it to the beginning or end of the task. Also, you can dictate what task should appear the next time the Task Builder is open, whether it be a new task, all tasks from the previous session or selected task from the previous session.

The Task Builder **General Options** page can be accessed by navigating to **File** > **Options** > **General**. General options is a subset of Task Builder Options.



Parameters

The following table describes Task Builder's available General options:

Property	Default	Description
On startup, open	New task	Dictates the Task Builder's startup behavior. This option is relevant only when the Task Builder is started manually from Windows Start menu or by the executable file by way of Program Files directory (for example, `C:\Program Files (x86)\Automate Desktop2024\AMTB.exe`). The previous task is never opened when Task Builder is started for a managed task through Task Administrator. • **New task** - A new task will open. Previously opened tasks are ignored. • **Selected task from last session** - Automatically re-opens the task that was last open in the Task Builder. If multiple tasks were previously open, the active/selected task is the one that will re-open. • **All tasks from last session** - Automatically re-opens all tasks that were previously open.

Property	Default	Description
When an action is selected	Add step to end of task	When double-clicking an action from the Actions panel, determines where that action is inserted in the Steps panel. The available options are: • **Add step to end of task** - The action is automatically inserted in the Steps as the last step. • **Insert step at current selection** - The action is automatically inserted in the Steps panel directly where the currently selected (highlighted) step is positioned. The current step and all others below it are moved a single step downward. **NOTE:** In either case, a new step can be inserted at the desired position by dragging the action from the Actions panel onto the desired position in the Steps panel instead of double-clicking it.

Property	Default	Description
Include error descriptions in Visual view	Only when different from default	Controls when the Visual view (as opposed to AML view) in the Steps panel displays descriptive text about each step's error handling along with descriptive text about the step itself. Every Automate Desktop step performs default error handling if the step fails. Select from the following: • **Always** - Always appends the error handling description to each step. This can cause extra, redundant text to be displayed in the Steps pane. • **Only when different from default** - Appends descriptive text about the step's error handling only when an error handling option has been set that is different from the default. This will cause the error description to be omitted in the majority of cases because the default error handling is sufficient for most steps (selected by default). • **Never** - Never appends the

Property	Default	Description
		error handling description. The error handling can only be viewed by opening the properties dialog for the step.
Task Builder auto save options	Disabled	If selected, the **Automatically save task(s) every 2 minutes** option will automatically save tasks with an associated file every 2 minutes. > **NOTE:** After selecting or clearing this setting, Task Builder must be restarted before changes will take effect.

Property	Default	Description
Default task file location	C:\Users\<user.name>\Documents\Automate Desktop2024 Tasks	By default, unmanaged tasks are saved to a location accessible by the user that initially installed the program. This location is different from the default managed task location and may vary depending on the operating system. To indicate the default location to which an unmanaged task should be saved, click the folder icon to browse available folders to select from or enter the full path to the folder in the provided text box. **NOTE:** If specifying a network location as the default task location, it is critical that mapped drive letters not be used. Folders should always be specified using UNC (Universal Naming Convention) paths. For example: `X:\pathname\` should be `\\servername\pathname\`. This is because mapped drives are available only when a user logs on and this does not occur for LocalSystem services.

Task Builder Formatting Options

Task Builder formatting preferences allow you to modify how activities that represent a block of steps, such as steps inside a Loop or If statement, are formatted in the Steps panel. You can choose to automatically add closing steps to those that require one (for example, Loop step should automatically be followed by an End Loop step) or you can choose to add closing steps manually as you construct the task. You can also decide upon whether those blocks of steps should be indented and what the indentation value should be.

To access the Formatting Options page, from Task Builder, select **File** -> **Options** -> **Debugger** -> **Formatting**. Formatting options is a subset of Task Builder Options.

Parameters

The following table describes Task Builder's available Formatting options.

Property	Default	Description
Wrap step text	Enabled	If enabled, the text that appears in the Steps panel will automatically wrap itself so as not to run off the right edge of the window. This enables all text to be viewed without the need to scroll to the right. If disabled, the text that appears in the Steps panel will run off the right edge.

Property	Default	Description
Step level auto-completion	Enabled	If enabled, any activity added to the Steps panel that starts a block of steps will be followed by the activity used to end the block. For example: • If enabled, any If statement will automatically be followed by an End If step. If disabled, you must manually add an End If step. • If enabled, any Loop step will automatically be followed by an End Loop step. If disabled, you must manually add an End Loop step. • If enabled, any Case statement will automatically be followed by an End Case step. If disabled, you must manually add an End Case step. > **NOTE:** An activity can be added from the Actions panel onto the Steps panel by way of a double-click or drag-and-drop operation. If using drag-and-drop to add your steps, setting the Task Builder General Option titled **When an action is selected** to **Add step to end of task** will make this option irrelevant because all activities that are added by way of double-click action will automatically be added as the last step of the task.
Automatically enclose proceeding step when starting a block	Disabled	If enabled, the proceeding step is automatically enclosed between the newly inserted block of steps. If disabled, no steps will be enclosed between the start and end block. This option is available only if the **Step level auto-completion** option is enabled.
Show visual effects while dragging	Enabled	If enabled, a visual overlay of the item being dragged along with a red line will be displayed in the Steps panel to determine the drop location. If disabled, only a red line will be displayed during a drag and drop operation.

Property	Default	Description
Animate steps while dragging	Enabled	If enabled, a drag and drop operation performed in the Steps panel will animate all affected steps to show the user how those steps would change position in response to the current operation. For example, when you drag an activity between two steps, they will move apart to make room for the drag source. If disabled, no animation will be applied to other steps during a drag and drop operation.
Shade steps within block	Enabled	If enabled, any steps contained between the start and end of a block of steps will be shaded. If disabled, no shading will be applied to any steps contained within a block of steps.
Auto-indent Steps	Enabled	If enabled, allows steps to be automatically indented. The default indent value can be entered in the **Indent Spacing Increment** parameter below.
Show Block Delimiter	Enabled	If enabled, a vertical line will be displayed to specify the start and end of a block of steps. If disabled, no delimiter will be shown. This option is active only if the **Auto-indent steps** option above is enabled. > **NOTE:** Delimiters only appear when the block of steps are in expanded view. They are hidden in collapsed view.
Indent Spacing Increment	7	Specifies the indentation value. This parameter is set to 5 by default and only available if the **Suggest Indent** parameter is enabled.

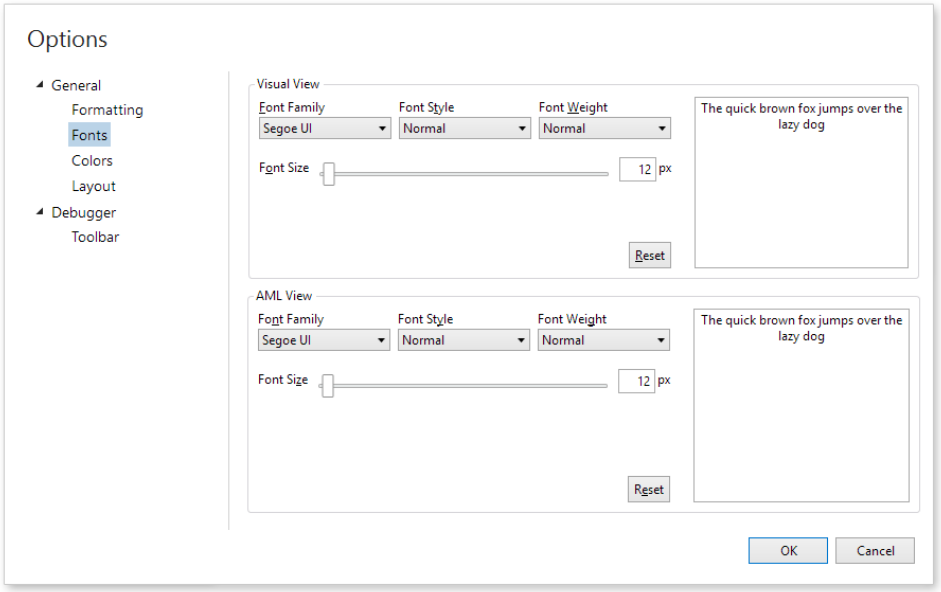
Step Visual View Parameters

Property	Default	Description
Show Action and Activity Names in step title	Disabled	If enable, each step in the Steps panel will display its designated action and activity name. If disabled, no action and activity name will be displayed. **NOTE:** If the action and activity name is not displayed, a step can still be recognized by its unique icon and inline description (if these options are enabled).
Show Inline Descriptions	Enabled	If enabled, an inline description of what the step will perform will be included. If disabled, no description will appear.

Task Builder Font Options

Task Builder font options allow customization of fonts (that is, font family, style, weight and size) to be displayed in the Steps Panel when the task is set to either Visual view or AML view.

To access Task Builder's Font Options page, navigate to **File** > **Options** > **General** > **Fonts**. Font options is a subset of Task Builder Options.



Parameters

The following table describes available font properties, as well as their Visual and AML defaults values. To reset all values to default, click the **Reset** button.

Property	Visual/AML	Description
Font Family	Segoe UI/Courier New	The name of the font to be used when Visual mode is selected from the View menu. An easy to read font for this view is suggested, such as Segoe UI or Tahoma, since Visual mode is meant to provide a clear, natural-language description about the task's steps.
Font Style	Normal/Normal	The style of font to be used. The available options are: • **Normal** - The text is shown normally. • **Italic** - The text is shown in italics. • **Oblique** - The text is "leaning" (oblique is very similar to italic, but less supported).
Font Weight	Normal/Normal	The weight of the font (or the thickness of the character outlines relative to their height). The available options are: • **Light** - Defines lighter or thinner characters. • **350** - Defines characters with weight class between **Light** and **Normal**. • **Normal** - Defines normal characters. • **SemiBold** - Defines semi-thick characters. • **Bold** - Defines thick characters. • **Black** - Defines thicker characters.

Property	Visual/AML	Description
Font Size	12 pixels/12 pixels	The font to be used when AML mode is selected from the View menu. A fixed-width font, such as Courier New, for AML view is suggested, since this type of view is typically used for fine, programmatic refinements to task steps.

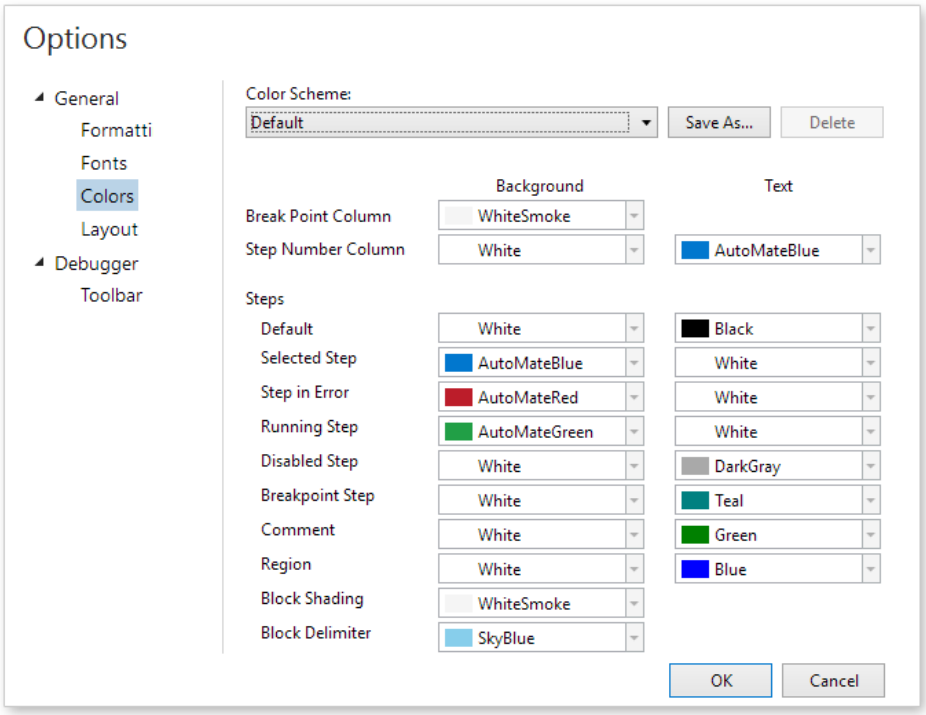
Task Builder Color Options

Task Builder's Steps panel can be configured to use a predefined color scheme or you can fully customize the collection of colors used to represent the steps of your task and save it as your own personal color scheme. A color scheme allows different users of Task Builder to create their own custom "look and feel" when creating, editing and debugging tasks. When saving a custom color scheme, that color scheme becomes your default scheme and is available only to you.

To access the color Options page, from Task Builder, select **File** > **Options** > **General** > **Colors**. Color options is a subset of Task Builder Options.

Parameters

The following table describes elements in the Steps panel that text and background colors apply to along with text and background colors that are set as default.


Options
General
Formatti
Fonts
Colors
Layout
Debugger
Toolbar
Color Scheme:
Default
Save As...
Delete
Background
Text
Break Point Column
WhiteSmoke
Step Number Column
White
AutoMateBlue
Steps
Default
White
Black
Selected Step
AutoMateBlue
White
Step in Error
AutoMateRed
White
Running Step
AutoMateGreen
White
Disabled Step
White
DarkGray
Breakpoint Step
White
Teal
Comment
White
Green
Region
White
Blue
Block Shading
WhiteSmoke
Block Delimiter
SkyBlue
OK
Cancel

Property	Background	Text	Description
Color Scheme	---	---	Indicates the combination of colors to be applied to text and background elements that appear in the Steps panel. To create your own custom color scheme, select a color for each element, click **Save As** and enter a name for the new scheme. To delete a custom color scheme, select it from the drop-down list and click **Delete**. You can also select from a list of pre-defined color schemes which are as follows: • **Default** - The default color scheme used in Automate Desktop 2024. • **Automate 10** - The default color scheme used in Automate 10. • **Automate 9** - The default color scheme used in Automate 9. • **Automate 6** - The default color scheme used in Automate 6. • **Automate 5** - The default color scheme used in Automate 5. • **Plain** - A color scheme made up of basic colors. **NOTE:** Only custom color schemes can be edited or deleted. To create a custom color scheme, you must first use the **Save As** option to save a copy of any original scheme.

Property	Background	Text	Description
Breakpoint Column	White Smoke	---	The background color of the column that contains Breakpoints. This is the left-most column in the Steps panel.
Step Number Column	White	Automate Blue	The text and background color of the column that displays step numbers. This column appears along the right side of the Breakpoints column, which is the left-most column in the Steps panel.
Default	White	Black	The text and background color for each default line of text that appears in the Steps panel. A step that does not fit the characteristics of any properties described below would be considered default.
Step Expression	---	Orange	The text color of a step expression.
Selected Step	Automate Blue	White	The text and background color of one or more steps that are currently selected (highlighted).
Step in Error	Red	White	The text and background color of a step that caused an error. When debugging a task from Task Builder, this color combination will appear in the Steps panel for a step that throws an error.
Running Step	Green	White	The text and background color of the step that's currently running. This color combination allows easy tracking of step by step execution while debugging a task.
Disabled Step	White	Gray	The text and background color of a disabled step. Disabled steps are not executed at runtime.

Property	Background	Text	Description
Breakpoint Step	White	Teal	The text and background color of any step that contains a Breakpoint. For debugging purposes, when a task is run from Task Builder, execution pauses at any step defined with a breakpoint.
Comment	White	Green	The text and background color of any step that contains a Comment. In Automate Desktop, comments provide a means for adding remarks or placing notes into your task. A comment is not an action or activity, therefore, it is skipped during runtime.
Region	White	Blue	The text and background color of any step that signifies the start and end of a Region. **NOTE:** This only applies to expanded regions. When a region is collapsed, the step that marks the start of the region only appears. The remainder of steps, including the one that marks the end of the region are hidden.
Block Shading	White Smoke	---	When selecting the first step in a block of steps (for example, Loop blocks, If Statements, Regions) this background color signifies the steps contained in the block, up to and including the last step. **NOTE:** This only applies to expanded regions. When a region is collapsed, the step that marks the start of the region only appears. The remainder of steps, including the one that marks the end of the region are hidden.

Property	Background	Text	Description
Block Delimiter	Sky Blue	---	The color of the delimiter line used to signify a block of steps, including the first and last step. **NOTE:** This only applies to expanded regions. When a region is collapsed, the step that marks the start of the region only appears. The remainder of steps, including the one that marks the end of the region are hidden.

To switch color schemes

1. In the **Color Scheme** parameter, click the drop- down combo box and select the desired scheme from the list that appears.
2. The selected scheme's name appears in the **Color Scheme** parameter and automatically becomes the default scheme.
3. Click **OK** to save changes.

To create a custom color scheme

1. Create a copy of the default color scheme or any existing scheme by selecting it from the **Color Scheme** parameter's drop-down list and clicking **Save As...**.
2. Enter a new name in the dialog that appears and click **OK**. The new scheme you created is automatically saved as the default color scheme.
3. Select the desired color for each text and background elements.
4. Click **OK** to save changes.

> **NOTE:** The default color scheme or any existing ones that appear in the **Color Scheme** parameter's drop-down list are not customizable. You must first create a copy of one of the existing schemes in order to customize it.

To delete a custom color scheme

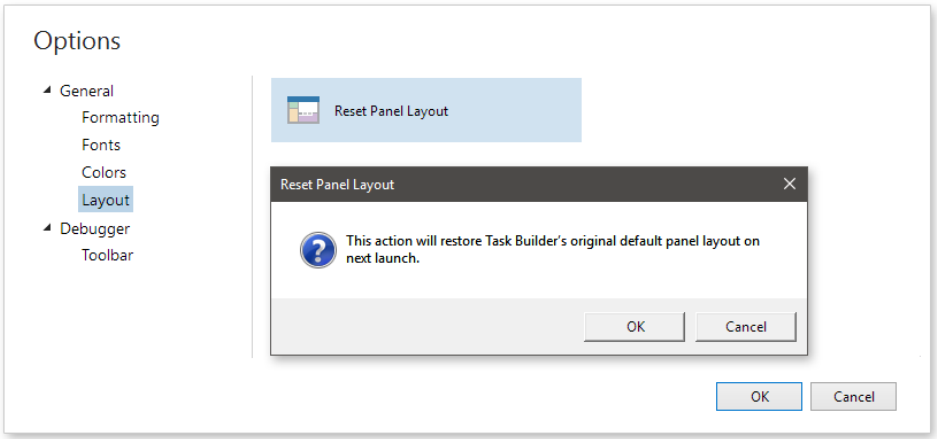
1. In the **Color Scheme** parameter, click the drop-down arrow and select the name of the custom scheme you wish to delete, then click **Delete**.
2. A dialog appears verifying whether or not to delete the custom color scheme. Click **Yes** to complete the operation.

> **NOTE:** The default color scheme or any existing pre-defined color schemes that appear in the **Color Scheme** parameter's drop-down list cannot be deleted. Only custom color schemes are allowed to be deleted.

Task Builder Layout Options

Task Builder's interface is customizable in ways that enable users to drag and dock certain panels wherever they please, unpin docked panels to enter an auto-hide state, and float panels separately. The Layout Options page lets you reset Task Builder's panels back to its original design on the next launch.

To access Task Builder Layout options, navigate to **File** > **Options** > **General** > **Layout**. Layout options is a subset of Task Builder Options.



To reset Task Builder the default layout

1. From the Layout section, click **Reset Panel Layout**.
2. A pop-up window informs you that Task Builder's original default panel layout will be restored on the next launch. Click **OK** to complete the operation. At this point, you must close all open instances of Task Builder. Thereafter, any new instance of Task Builder will display its original panel layout.

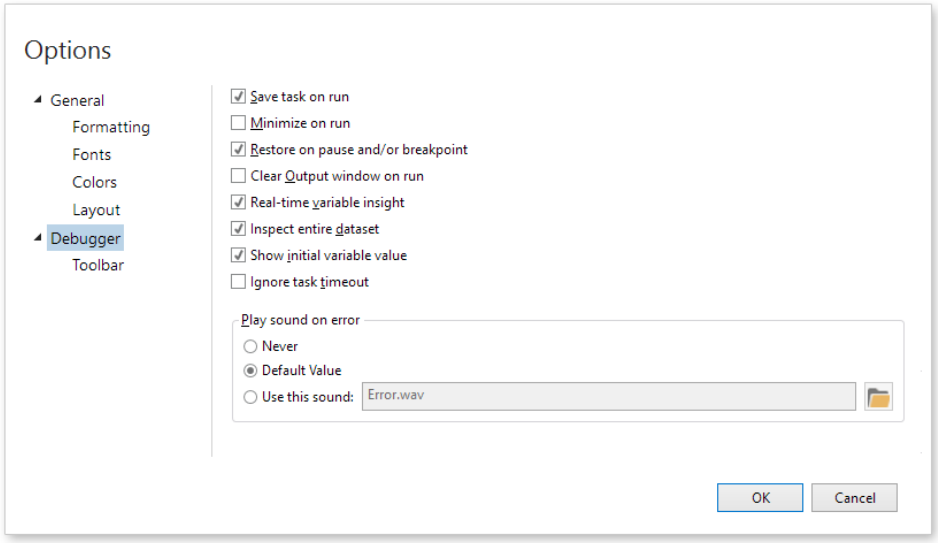
> **NOTE:** If you change your mind, you can always cancel the reset layout operation before closing Task Builder by returning to the Layout section and clicking **Cancel Reset Panel Layout.**

Task Builder Debugger Options

The Task Builder includes settings that affect its behavior while running a task. In some cases you may want to change these settings to better fit your debugging needs. For instance, you can opt to minimize the Task Builder window during execution to better examine an interactive task, automatically clear the Output Debug Panel after each run, or play a custom sound when an error occurs.

To access Task Builder's Debugger Options page

- Click to **File> Options>Debugger**. Debugger options is a subset of Task Builder Options.



Parameters

The following table describes available Debugger options:

Property	Default	Description
Save task on run	Disabled	If enabled, Task Builder automatically saves changes made to a task when clicking the **Run** button. If the task has not been saved yet, Automate Desktop will prompt the user whether or not to save before running. If disabled, changes will not be saved upon run.
Minimize on run	Disabled	If enabled, Task Builder minimizes to the system tray when execution starts. When the task ends, Task Builder restores itself onto the desktop and highlights the current step or displays **Finished** as the step status in the status bar. If disabled, Task Builder stays on the desktop (does not minimize or hide itself) during task execution. > **NOTE:** It is recommended to enable this option when testing tasks that include interactive steps (for example, Send Keystrokes, Move Mouse to Object actions) or steps that interact with other windows (for example, Maximize Window, Minimize Window actions). This is due to leaving the Task Builder window on the desktop may interfere with other windows that appear during runtime. Doing so may result in a Send Keystrokes step, for example, failing to send keystrokes to the proper window or application.
Restore on pause and/or breakpoint	Enabled	If enabled, when task execution is manually paused (when a user clicks the **Pause** button) or pauses at a breakpoint, and the Task Builder window is currently minimized, it restores itself and highlights the current step. When execution continues, the Task Builder window, once again, minimizes to the system tray.

Property	Default	Description
Clear Output window on run	Enabled	If enabled, clears the contents of the Output window before each task execution . If disabled, new debugging output will append existing output accumulated from previous runs.
Limit entries in output panel to	Disabled	If enabled, limits the number of entries in the Output panel to the numeric value entered here. Once the value is exceeded, the oldest entry is deleted. If disabled, entries are not limited.
Real-time variable insight	Enabled	If enabled, the Variables window is updated after each step to show the current value of existing variables on a step by step basis. If disabled, the Variables window is updated only when the task ends or when it is in a paused state (by clicking the **Pause** button, when a Breakpoint is encountered or while stepping through with the use of the **Step** button). **WARNING:** Enabling this option could cause Task Builder to run tasks slower.
Inspect entire dataset	Disabled	If enabled, during runtime, extended dataset properties that are normally hidden (for example, creation date/time, total rows) will be displayed in the Variable window. If disabled, only basic dataset properties will be displayed. **WARNING:** Enabling this option could cause Task Builder to run tasks slower.
Show initial variable value	Enabled	If enabled, causes the Variables window to include a column for **Initial Value** in addition to the **Current Value** column. If disabled, only the current value is shown.

Property	Default	Description
Play sound on error	Error.wav	Denotes whether to play a sound when the task stops as a result of an error. The selections are: • **Never** - Do not play a sound when an error occurs. • **Use default sound** - Use the default system error sound, as specified in Window's Control Panel. • **Use this sound** - Specify a valid path and file name of a .wav file to use.

Task Builder Ribbon

The ribbon is a contextual interface designed to increase efficiency and make it easier for users to find features and quickly access popular commands and controls during task construction. The ribbon spans the top of Task Builder, directly below the title bar. It is filled with graphical representations of control elements which are grouped by different functionality. Commands are organized in logical groups, which are collected together under operation related tabs. The ribbon brings the most common commands to the forefront, in plain sight, so you no longer have to roam aimlessly through the clutter of menus, sub-menus, and toolbars searching for what you want.

Enlarging or maximizing the Task Builder window auto-expands the ribbon vertically. To add additional space, the ribbon can be collapsed so only the tabs appear. This can be accomplished by clicking the **Pin/Unpin Ribbon** button. Since not every single option will fit on the ribbon, drop-down menus are available which are represented by small arrows below certain buttons/commands. Clicking such arrows displays a drop-down menu which contain related sub-commands. For example, if you click the arrow below the **Run** icon, a drop-down menu will appear displaying related commands such as **Run All**, **Run Selected**, **Run From Here**, and **Step** (shown below).

Tabs and Commands

The Task Builder ribbon is categorized into three contextual tabs, **File**, **Home**, and **Actions**. Each represents an activity area grouped together with related commands, controls or buttons that represents the operation to perform. To view a description of each command

and associated hot-keys (if any), hover the mouse cursor directly above the desired command. The following tables describe the available commands contained within each tab.

Home Tab Commands

The following table details each command found in the ribbon's **Home** tab along with their associated controls and keyboard hot-key combinations (if applicable). Each command is categorized by the group they reside in.

Group	Command	Description	Hot-Key
Document	Save and Close	Saves the current task and closes the Task Builder window.	ALT+ SHIFT + A
	Save Copy As...	Saves a copy of the task as an AML (Automate Desktop Markup Language) file.	CTRL + SHIFT + S

Group	Command	Description	Hot-Key
Clipboard	Cut	Cuts the selected steps. To select multiple steps, hold down CTRL during selection. To select all steps, press CTRL+A or click from the Find group. More on copying reusing task steps	CTRL + X
	Copy	Copies the selected steps to the clipboard. To select multiple steps, hold down CTRL during selection. More on copying & reusing task steps	CTRL + C
	Copy Description	Copies the text description of the selected steps. The description can then be pasted into another document such as Notepad. To select multiple steps, hold down CTRL during selection. More on copying & reusing task steps	CTRL + D
	Paste	Pastes the previously copied or cut steps. The stepss are pasted directly below the currently highlighted line in the Steps pane. More on copying & reusing task steps	CTRL + V
	Delete	Deletes the selected steps. To select multiple steps, hold down CTRL during selection.	(DEL)

Group	Command	Description	Hot-Key
Find	Find	Finds a specific word or phrase within the task. More on Find & Replace	CTRL + F
	Find Next	Finds the next instance of a word or phrase. More on Find & Replace	F11
	Find Previous	Finds the previous instance of a word or phrase. More on Find & Replace	F12
	Replace	Replaces an occurrence of the selected text string. More on Find & Replace	CTRL + H
	Replace Next	Replaces the next occurrence of the selected text string. More on Find & Replace	CTRL + SHIFT +R
	Select All	Selects all steps included in the Steps panel.	CTRL + A
	Select Step Number	Opens a dialog allowing entry of the desired step to select. Mainly used to quickly navigate to a specific step within a task that includes a significantly large amount of steps.	CTRL + G
	Select Label	Opens a dialog allowing entry of the desired label to go to. Used to jump to another section of the task defined by a Label.	CTRL + L

Group	Command	Description	Hot-Key
Layout	Step numbers	Turns step numbers On/Off as they appear on the Steps pane (On by default). More on editing task steps	---
	Word Wrap	Turns word wrap On/Off. When set to ON, the text becomes wrapped in the Steps pane so it doesn't scroll across the entire screen (On by default). More on editing task steps	---
	Panel Layout	Shows/Hides specific Task Builder windows. All windows are shown by default.	---
Task	Run	Runs the steps of the current task sequentially. More on run options	F5
	Run Selected	Runs the selected steps. More on run options	CTRL + F5
	Run from Here	Runs all steps from the specified step onward. More on run options	SHIFT + CTRL + F5
	Step	Runs the task one step at a time starting from the highlighted step. Click this button each time you want to run the next step. More on run options	CTRL + F2
	Pause	Pauses a running task. More on run options	F6
	Stop	Stops and resets a running task. More on run options	CTRL + F2

Group	Command	Description	Hot-Key
Step	Edit	Opens the properties of the selected step for editing.	---
	Up Arrow	Moves the selected steps up one line. To select multiple steps, hold down CTRL during selection. More on editing task steps	CTRL + UP
	Down Arrow	Moves the selected steps down one line. To select multiple steps, hold down CTRL during selection. More on editing task steps	CTRL + DOWN
	Left Arrow	Decreases indentation of the selected steps. To select multiple steps, hold down CTRL during selection. More on editing task steps	CTRL + LEFT
	Right Arrow	Increases indentation of the selected steps. To select multiple steps, hold down CTRL during selection. More on editing task steps	CTRL + RIGHT
	Create Region	Creates a region. More on Regions	CTRL + R
	Encapsulate in Region	Encapsulates any selected steps into a region. More on Regions	CTRL + E
	Rename Region	Renames the selected region. More on Regions	---
	Clear Region	Clears the selected region. More on Regions	---
	Clear All Regions	Clears all regions. More on Regions	---
	Expand All Regions	Expands all regions. More on Regions	CTRL + SHIFT + E

Group	Command	Description	Hot-Key
	Collapse All Regions	Collapses all regions. More on Regions	CTRL + SHIFT + C
	Reset Indent	Removes indentation from a step. More on editing task steps	---
	Disable	Enables/Disables execution of the selected steps. To select multiple steps, hold down CTRL during selection. Disabled steps appear grayed out and are ignored during task execution. Newly added steps are enabled by default.	---
	Bookmark	Adds a bookmark to the selected steps. To add to multiple steps, hold down CTRL during selection. Bookmarks are a means of marking steps so you can find them easily when constructing or editing a task. More on bookmarks	CTRL + F3
	Next Bookmark	Goes to the next step that contains a bookmark. More on bookmarks	F3
	Previous Bookmark	Goes to the previous step that contains a bookmark. More on bookmarks	SHIFT + F3

Group	Command	Description	Hot-Key
View	Breakpoint	Adds a breakpoint to the selected steps. To add a breakpoint to multiple steps, hold down CTRL during selection. A breakpoint is a debug tool that pauses task execution at the specific step to allow the developer time to examine particular data that may change during the course of the task (for example, variables or datasets). More on Breakpoints	F8
	Visual Mode	The steps of the task are displayed in Visual Mode which displays a plain English text description of each step. More on editing task steps	---
	AML Mode	The steps of the task are displayed in AML (Automate Desktop Markup Language) mode, the internal language used by Automate Desktop. More on editing task steps	---
Insert	Add Variable	Adds a variable to the highlighted section of the Steps pane. More on Variables	CTRL + ALT + V
	Add Comment	Adds a comment to the highlighted section of the Steps pane. More on Comments	CTRL + ALT + C
	Add Attachment	Adds an attachment to the highlighted section of the Steps pane. More on Attachments	---
	Add Label	Adds a label to the highlighted section of the Steps pane. More on Labels	---

Group	Command	Description	Hot-Key
Recorder	New Recording	Opens the Automate Desktop**Recorder** – a powerful feature that simplifies the process of automating and recording user interface interactions and actions in Automate Desktop. By using the Recorder's intuitive interface, you can quickly and easily create and add recordings to new or existing tasks. More on the Recorder	CTRL + ALT + R
	Edit	Edits an existing Recorder recording. More on the Recorder	CTRL+ ALT + E
	Split	Splits a Recorder recording into two recordings at the specified step. More on the Recorder	CTRL + ALT + T

Actions Tab Commands

The following table details each command found in the ribbon's **Available Actions** tab along with their associated controls and keyboard hot-key combinations (if applicable). Each command is categorized by the group they reside in.

Group	Command	Description
Actions Panel	Add Step	Gets the action selected from the Available Actions pane and places it as the last step on the Steps pane. More on Steps
	Insert Step	Gets the action selected from the Available Actions pane and places it in the specified location on the Steps pane. More on Steps
	Expand All	Expands all folders in the Available Actions pane. More on Available actions Pane
	Collapse All	Collapses all folders in the Available Actions pane. More on Available Actions Pane
	Categories	Enables/Disables categorized folders in the Available Actions Toolkit (Enabled by default). More on Available actions Pane
	Sort A to Z	Changes the sort order of the current Available Actions view to alphabetical A to Z. Click again to reverse the sort order. More on Available Actions Pane

Group	Command	Description
My Actions Panel	Clear Favorites	Clears the My Actions favorite actions list. More on My Actions pane
	Clear Recent	Clears the My Actions recent list of actions. More on My Actions pane
	Clear All	Clears all the My Actions Favorites, Snippets, Recent, and Frequent lists. More on My Actions pane
	Clear Snippets	Clears the My Actions snippet action list. More on My Actions pane
	Clear Frequent	Clears the My Actions frequent actions list. More on My Actions pane
	Clear Folder	Clears the selected snippet folder in the My Actions pane. More on My Actions pane
	Add to Favorites	Adds the selected action to the Favorites list. More on My Actions pane
	Add as Snippet	Adds the selected step as a snippet. More on My Actions pane
	Create Folder	Creates a snippets folder. More on My Actions pane
	Remove Selected	Removes the selected step, snippet, or region. More on My Actions pane

File Menu

The **File** tab is located in the upper-left corner of the ribbon and presents a menu of commands and operations that do not directly relate to the editing of the task at hand. They include items that normally reside in the **File** menu of legacy systems, such as **New**, **Open**, **Save**, **Print**, and **Close** . The right-hand portion of the menu lists the most recently opened tasks. These are always conveniently visible so that you don't have to search your computer for a task you previously opened or frequently work on. The bottom right corner of the menu contains the **Options** button. Clicking this button opens a **Preferences** dialog that supplies

an assortment options used to configure visual and behavioral attributes of the Task Builder.

File Menu Commands

The following table provides a description for each command found when clicking the **Application** button along with their associated controls and keyboard hot-keys (if applicable).

Command	Description	Hot-Key
Help	Opens the embedded help files.	---
New	Creates a new task. Selecting this option adds a new tab in the Steps pane. **NOTE:** Using this option creates an unmanaged task.	CTRL + N
Open...	Opens an existing task. A standard Windows Explorer dialog appears allowing you to navigate to the task (.AML) file to open.	CTRL + O
Save	Saves the current task. If the task being saved is a managed task, it will be saved to its present location. If saving an unmanaged task (for example, created using the **New** option), a standard Windows Explorer dialog appears allowing you to select the folder in which to save the task (.AML) file.	CTRL + S
Save As	Saves a copy of the task file or saves the task file in another format. A standard Windows Explorer dialog appears allowing you to navigate to the location in which to save the task (.AML) file.	SHIFT + CTRL + S
Save All	Save all open tasks.	---
Close	Closes the current task file. If changes were made, a dialog will appear prompting whether to save the changes. Click **Yes** to proceed.	---

Command	Description	Hot-Key
Close All	Closes all currently opened tasks. If changes were made to one or more tasks, a dialog will appear prompting whether to save the changes. Click **Yes** to proceed.	---
Print	Prints the steps of the current task as they appear in the Steps pane.	---
Send	Emails the current task to the specified recipient. The available send options are: • **Mail Recipient** - Sends the displayed task in the body of an email. • **Mail Recipient (As Attachment) -** Sends the displayed task as an email attachment. • **Mail All to Recipient (As Attachment) -**Sends all open tasks as email attachments.	---
Close	Closes the current task file. If changes were made, a dialog will appear prompting whether to save the changes. Click **Yes** to proceed.	---
Close All	Closes all currently opened tasks. If changes were made to one or more tasks, a dialog will appear prompting whether to save the changes. Click **Yes** to proceed.	---
Options	Allows access to available Task Builder Options. The table below lists available options.	---
Exit	Closes the Task Builder.	---

Title Bar

The title bar spans across the top of Task Builder. It contains the title of the current task as well as controls that enable you to set the Task Builder interface to full screen or

minimize/maximize the current display. By default, it also contains the Quick Access Toolbar which provides easy one-click access to commonly used controls.

Quick Access Toolbar

The Quick Access toolbar is a small, customizable area to the upper left of the ribbon that provides you with one-click shortcuts to commonly used commands that are independent of the ribbon tabs that are currently displayed. You can add specific commands to the Quick Access Toolbar so they are available no matter which tab you are on. Additionally, you can move the toolbar from one of two possible locations; above or below the ribbon, or you can hide it altogether to save space.

Quick Access Toolbar Default Commands

By default, the Quick Access toolbar exposes single-click buttons that represent the most often used commands, such as **Open**, **Save**, **Undo,Redo**, and **Run.** The following table provides a description of those commands:

Command	Description	Hot-Key
Open	Opens an existing task. A standard Windows Explorer dialog appears allowing you to navigate to the task (.AML) file to open.	CTRL + O
Save	Saves the current task. If the task being saved is a managed task, it will be saved to its present location. If saving an unmanaged task (for example, created using the **New** option), a standard Windows Explorer dialog appears allowing you to select the folder in which to save the task (.AML) file.	CTRL + S
Find	Finds a specific word or phrase. Sub-commands are as follows: • **Find Next** - Finds the next instance of a word or phrase. • **Find Previous** - Finds the previous instance of a word or phrase.	---

Command	Description	Hot-Key
Run	Runs the steps of the current task sequentially. Sub-commands are as follows: • **Run All** - Runs all the steps of the current task sequentially. • **Run Selected**- Runs the selected steps. • **Run From Here**- Runs the task from the selected step. • **Step**- Runs the task step by step.	F5
Undo	Returns to a previous state by undoing the effects of one or more commands. Helpful for undoing mistakes.	CTRL + Z
Redo	Returns back to the state that was previously undone.	CTRL + Y

To add a command to the Quick Access toolbar

1. On the ribbon, click the appropriate tab or group to display the command that you want to add to the Quick Access Toolbar.
2. Right-click the command, and then click **Add to Quick Access Toolbar** on the shortcut menu.

To remove a command from the Quick Access toolbar

1. Right-click the command you want to remove from the Quick Access Toolbar.
2. Click **Remove from Quick Access Toolbar** on the menu that appears.

To remove a default command from the Quick Access toolbar

1. Click the **More Buttons** icon (to the right of the toolbar).
2. On the drop-down menu that appears, select the commands you want to omit from the toolbar. The check mark beside that command will disappear indicating that the command has been removed.

3. To reinstate the command, follow the above directions and simply select the command previously omitted.

To display the Quick Access toolbar below the ribbon

1. Click the **More Buttons** icon (to the right of the toolbar).
2. From the drop-down menu, select the option **Show Quick Access Toolbar Below the Ribbon**. The toolbar is then displayed below the ribbon.
3. To reinstate the toolbar above the ribbon, select the option **Show Quick Access Toolbar Above the Ribbon.**

Access Keys

If you prefer to use the keyboard instead of the mouse, the ribbon provides keyboard shortcuts called Access Keys that enable you to quickly perform operations without reaching for the mouse. Access keys provide a quick and convenient way to use a command by pressing a few keystrokes, no matter where you are in the program.

To use the access keys

1. Press and release the ALT key. The KeyTips are displayed over each feature that is available in the current view.
2. Press the letter, number or character combination shown in the KeyTip over the feature that you want to use. Depending on which letter/number you press, you may be shown additional KeyTips.
3. Continue pressing letters/numbers until you reach the command or option that you want to use.

Other Keyboard Navigation Options

Another way to use the keyboard to work with the ribbon is to move the focus among the tabs and commands until you find the feature that you want to use. The following table lists some ways to move the keyboard focus without using the mouse:

To Do This...	
Select the active tab of the ribbon and activate the access keys.	ALT or F10. Press either of these keys again to move back to the document and cancel the access keys.
Move to another tab of the ribbon.	ALT or F10 to select the active tab, and then LEFT ARROW or RIGHT ARROW
Minimize or restore the ribbon.	CTRL + F1
Move the focus to each command in the ribbon, forward or backward.	ALT or F10, and then TAB or SHIFT + TAB
Move down, up, left, or right among the items in the ribbon.	DOWN ARROW, UP ARROW, LEFT ARROW, or RIGHT ARROW
Activate the selected command or control in the ribbon.	SPACE BAR or ENTER
Open the selected menu or gallery in the ribbon.	SPACE BAR or ENTER

Task Builder Status Bar

The area at the bottom of the Task Builder comprises an intuitive status bar that displays real-time progress of a running task through graphics and text. When a task is run from the Task Builder, the status bar graphically indicates execution state by means of a progress indicator and activity icons. Textual information such as execution status and runtime details are also displayed. The status bar is a great way to provide useful and relevant information without interrupting execution or breaking task flow, which can be very helpful and convenient for task developers, typically during testing periods.

Status Bar Controls

The status bar is divided into sections, each of which displays particular information about the running task. The various elements are detailed below:

Control	Description
Running step/Total steps	Displays the current step being executed as well as the total number of steps the task contains.
Execution status	Displays the status of the currently running task. For example: • **Running** - Task is currently in a running state. • **Completed** - Task completed. • **Error** - Task generated an error.
Progress Bar	Displays total progress of the current step being executed.
Running step action icon	Displays the activity icon associated with the step currently running.
Step execution details	Displays information about the activity that's currently running. This section updates step by step information about the task during execution.

Control	Description
Zoom slider	The zoom slider allows you to increase or decrease the size of objects that appear in the Steps panel. The zoom value is displayed by percentage smaller/bigger than original size. This feature can be used in the following ways: • Move the slider left or right until the desired size is reached. • Click **+** or **-** to zoom in or out in increments. • Right-click the displayed percentage value and select from a drop-down list of predefined values. > **NOTE:** To revert back to the default percentage value (100%), left-click the current value that appears.
Zoom level	The zoom level percentage. To revert back to the default percentage value (100%), left-click the current value that appears here.

Task Builder Steps Panel

The Task Builder's Steps panel displays the actions/activities that are selected from the Actions panel that will be carried out when the task executes. During task construction, actions and activities are dragged from the Actions panel onto the Steps panel. During this process, a dialog displaying the properties of the selected action/activity opens and specific parameters and requirements can be viewed or modified. Once parameters are set, the action/activity and its properties are then displayed in the Steps panel and the action becomes a task "step". These steps run automatically in a sequential order when the task is triggered by some event or condition or when started manually by the user.

A single instance of the Task Builder can open multiple tasks, each separated by tabs located at the bottom of the Steps pane. Each tab is labeled with the name of the corresponding task. The steps of the desired task can be viewed by simply selecting the proper tab. When a task is selected, the Steps pane becomes populated with steps of that task and the Status Bar and Debug Pane show data for that task.

Task Builder's Steps panel can be configured to use a predefined color scheme or you can fully customize the collection of colors used to represent the steps of your task and save it as your own personal color scheme.

By default, a task step is comprised of the following:

- Step number
- Unique icon associated with the action
- Visual or AML description of the step (depending on which view is selected from the Ribbon's **View** command group).

Alternatively, step numbers can be omitted, icon sizes can be modified or step descriptions can be fully customized.

> **NOTE:** If you import a task that includes an unlicensed action/activity, the Steps panel will display the unlicensed step (in **Visual** view) as currently unlicensed. Running a task with an unlicensed step will always fail at that step. You can bypass the step by disabling it or removing it altogether.

Task Functions/Variables Panel

In order to make the creation and maintainability of tasks easier and safer, Automate Desktop now supports variable scoping, variable accessibility as well as external tasks and function declarations and execution. The top portion of the Steps pane contains the Task Functions & Task Variables Panel.

This drop-down panel is used to create Task Functions and Variables and display their properties in a clear and concise manner so the user understands the steps being displayed at any given time are associated with the function/variable being modified.

Context Menu

Right-clicking one or more lines of code on the Steps pane opens a pop-up menu that displays a list of available options you can perform for those steps. The table below contains a list of available Steps pane right-click menu items.

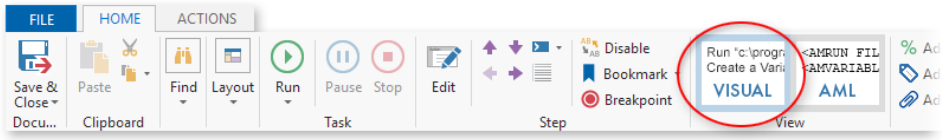
Menu Item	Description	Hot-Key
Cut	Cuts the selected steps. To select multiple steps, hold down CTRL during selection. To select all steps, press CTRL+A or click theSelect Allbutton from theFindcommand group of the Ribbon'sHometab.	CTRL + X
Copy	Copies the selected steps to the clipboard. To select multiple steps, hold down CTRL during selection. To select all steps, press CTRL+A or click theSelect Allbutton from theFindcommand group of the Ribbon'sHometab. Copied steps can be pasted in the Steps pane of the current task, another task or an application such as Notepad.	CTRL + C
Copy Description	Copies the description of the as they appear in Visual view	---
Paste	Pastes the previously copied or cut steps. Items are pasted directly below the currently highlighted line in the Steps panel.	CTRL + V
Delete	Deletes the selected steps. To select multiple steps, hold down CTRL during selection.	DEL
Select All	Selects all steps contained in the Steps panel.	CTRL + A
Select Step Number	Opens a *Select Step Number* dialog prompting you to enter the step number to select.	CTRL + G

Menu Item	Description	Hot-Key
Select Label	Opens a *Go to Label* dialog prompting you to enter the label to go to.	CTRL + L
Enable	Disables/enables the selected steps. A disabled step will not execute at runtime.	---
Edit Step	Edits the selected or highlighted step. Only a single step can be edited at a time.	---
Clear Regions	Clears the selected or highlighted regions.	---
Clear All Regions	Clears all regions contained in the current task. Note that this will clear all existing regions whether or not they are selected or highlighted.	---
Move Steps Up	Moves the selected steps one line up.	---
Move Steps Down	Moves the selected steps one line down.	---
Increase Indent	Increases indentation of the selected steps.	CTRL+ RIGHT
Decrease Indent	Decreases indentation of the selected steps.	CTRL+ LEFT
Add to Watch List	Adds the step to the Watch List. You can view watch list items by way of the Watches debug pane.	---
Add to Favorites	Adds the selected action to your Favorites list. Favorites as well as most often used and recently used actions can be viewed from the My Actions panel.	---

Custom Step Description

Typically, when a user finishes editing the parameters of an activity, a general description of that activity is displayed in Task Builder's Steps panel, assuming the **View** parameter is

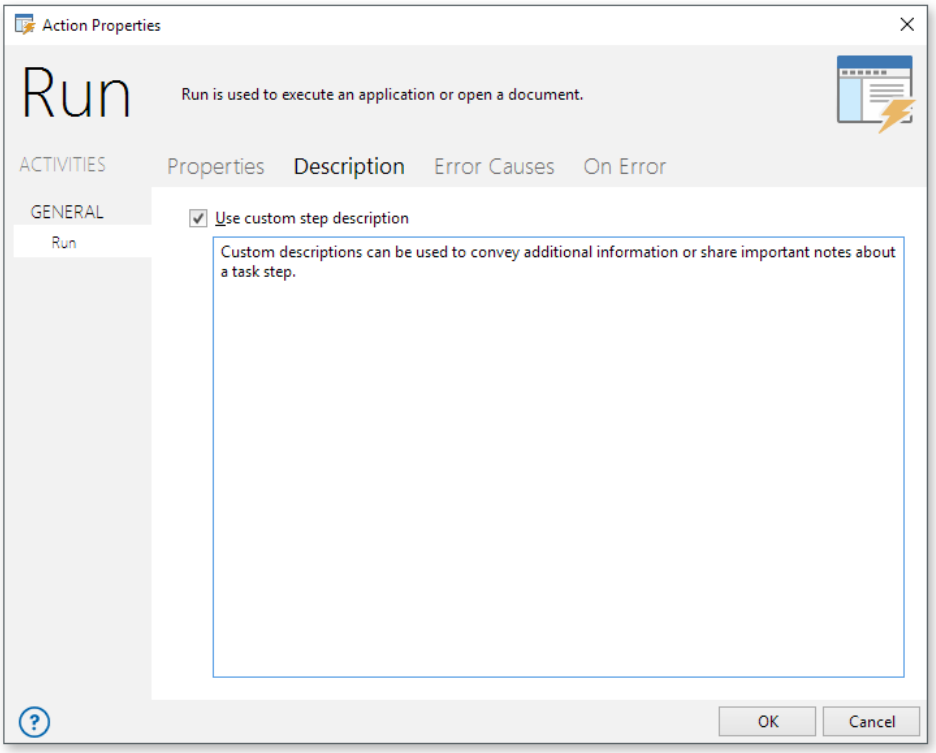
currently set to **Visual** mode (circled below). If needed, the default description can be modified or fully replaced with a custom description by way of the **Description** tab properties. This feature can be used to convey additional information about the activity or display special notes or instructions.



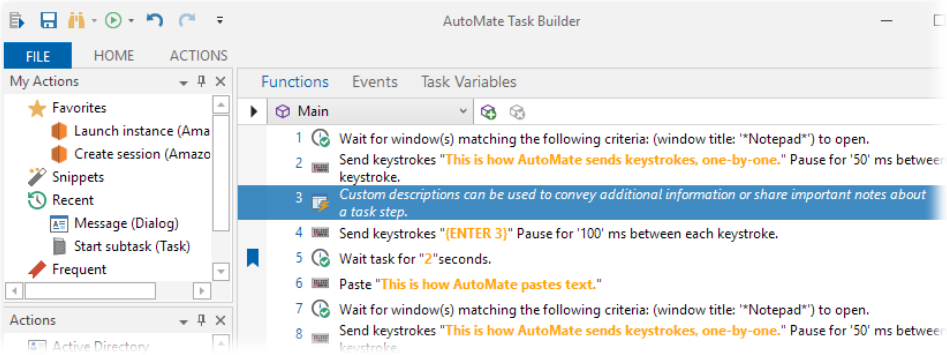
> **NOTE:** Custom step description is only supported if the Steps panel view is set to **Visual** mode, which displays the steps in plain English, and not AML mode, which displays the steps in AML (Automate Markup Language) format.

To enter a custom description

1. Open the properties of the activity/step, and then select the **Description** tab.
2. Enable the **Use custom step description** parameter.
3. Enter a desired description in the provided field (as illustrated below), and then select **OK**.



4. The Steps panel displays the custom description entered in place of the original description (as highlighted below).



NOTE:
Another way to document instructions, notes or other important information directly onto the Steps panel is by adding Comments. If you import a task that includes an unlicensed action / activity, whether or not it contains a custom description, the Steps panel will always display the unlicensed step (in **Visual** view) as currently unlicensed. Running a task with an unlicensed step will always fail at that step. You can bypass the step by disabling it or removing it altogether.

Percent Signs in Advanced Workflows

A percent sign is used as a special character to indicate the beginning and end of an expression. This allows variables, constants and other expressions to be entered in any parameter of a task's properties that accepts expressions. For example: %1+1% inside a task will resolve to 2 at runtime. Below are some common scenarios which involve the use percent signs.

Sending Literal Percent Sign

To send a literal percent sign, escape the percent character. Doubling the percent sign is known as "escaping" the percent sign. For example, to send the literal text 5%, specify 5%%. At runtime the double percentage signs will be recognized and converted into one.

All percent signs are recognized and evaluated by Advanced Workflows that are contained in any parameter that accepts expressions. For example, if an 'Open Webpage' action contained the URL:

https://m

During runtime, Advanced Workflows will detect the percent signs contained in the URL (%admin%) and automatically classify admin as an expression, which eventually fails the task with the error "A variable in the expression does not exist or is misspelled". In such cases, doubling the percent signs will resolve the issue as shown below:

https://

Concatenating Two Values in an Expression

In some cases, it is necessary to concatenate two variables, functions, or some combination thereof. In this case it is critical to remember that two percent signs always mean a literal percent. So the following will NOT work:

%variable1%%variable2%

(INCORRECT)

At runtime, Advanced Workflows would actually see the expression as variable1%variable2 which would result in a Syntax Error. The proper way to combine two values in an expression is to use the ampersand (&) character as follows:

%variable1 & variable2%

(CORRECT)

Common Misconception

A common misconception is that percentage signs are required anytime a variable is used. This is not always the case because percentages surround expressions which may or may not contain variables. For example, in situations where Advanced Workflows requires a variable name (e.g., in the 'Set Variable' action), only the variable name is required. It is permissible, however, to use an expression to specify the variable name so the resolved variable is used.

For example, assume we created a named Var1 with an initial value of Scott. If we wanted to change the value of this variable from Scott to Muscle, we would use a Set Variable action with the properties shown below.

If we add another variable named Var2 and set its initial value to Var1, we could set the Var1 variable to Muscle as we did above by using the expression shown below.

This is because at runtime the %Var2% expression will be evaluated to Var1 before the Set Variable action is executed. The code that executes this example is below.

<AMVARIABLE NAME=Var1Scott/

<AMVARIABLE NAME=Var2Var1/

<AMSET

> **NOTE:** The code can be copied and pasted directly into the Steps pane of the Task Builder.

Testing Tasks Using Run Options

Task execution may happen so rapidly that it is difficult for a user to determine the exact cause of a problematic step. In such cases, the Task Builder encompasses various **Run** options (displayed below) to handle this issue. The **Run Selected** control runs only the selected steps, which is a useful way to ignore execution of steps that are non-problematic. The **Run From Here** control runs the task from the selected step onward. The **Step** control suspends task execution on a step by step basis, allowing you to examine what the task is doing during execution of each step. This is ideal for troubleshooting problematic purposes because it lets you suspend execution at each step as well as view runtime information populated within the various Debug panel views.

> **NOTE:** When suspended, the task is still running but paused between actions. Automate Desktop variables or datasets retain their values, and the Task Builder's Output panel displays information about the step that is currently running.

Run Controls

The following table describes the Task Builder's list of available Run controls along with their associated icon and equivalent hot-key.

Icon	Name	Description	Hot-key
	Run/Run All	Runs the entire task, from beginning to end, or until an error is encountered, a breakpoint is reached, or the **Pause** button is manually clicked. If the task is manually paused or a breakpoint is encountered, execution is suspended indefinitely at the point of the pause or breakpoint. Use other available run options described below to continue execution.	F5
	Run Selected	Runs only the selected (highlighted) steps sequentially, from the lowest to the highest step. To select multiple steps, hold down CTRL during selection.	CTRL + F5

Icon	Name	Description	Hot-key
	Run from Here	Runs the task from the selected (highlighted) step until its completion, or until an error is encountered, a breakpoint is reached, or the **Pause** button is manually clicked.	SHIFT + CTRL + F5
	Step	Runs the task step by step, pausing after each step to allow enough time to examine the information displayed in the Debug panel. To continue to the next step, click **Step** or press **F9**. To run the remaining steps at normal speed, click **Continue** or **Run from Here**. **NOTE:** • Stepping can also be used to step through a task from a breakpoint or paused step. • When execution is paused, task state is suspended, however, functions, variables, and other debugging elements remain in memory. In order to fully stop task execution and reset debugging elements, click **Stop** or press **CTRL+F2**.	F9
	Continue	Continues execution of a paused task from the point of suspension until its completion, or until an error is encountered, a breakpoint is reached, or the **Pause** button is manually clicked.	F5
	Pause	Immediately and indefinitely suspends task execution at a particular step. Click **Continue** to resume execution or click **Step** to run remaining steps one by one.	F6
	Stop	Immediately stops the running task and resets all debugging elements.	CTRL + F2

Run Instructions

To run the entire task

Click the **Run All** button or press F5. This will run all of the steps of the task sequentially from beginning to end (or until an error is encountered, a pause is manually applied, or a breakpoint is reached).

To run one or more selected steps

1. Select the steps you wish to run from the Steps panel. To select more than one step, hold down CTRL during selection.
2. Click **Run Selected** or press CTRL+F5. This will run the specified steps in sequence from the lowest to highest numbered step.

To run the task from a specific step

1. Select the step that should start the execution process.
2. Click **Run From Here** or press SHIFT+CTRL+F5. The steps will run sequentially starting from the step you selected.

To step through a task

1. Click the **Step** button. The first (or selected) step of the task runs. At this time you can:
    a. Examine the contents of your variables by looking at the Variable View of the Debug Window.
    b. Examine the states of any watches you have set, using the Watches View of the Debug Window (see below for more information on watches).
    c. Observe what the previous step may (or may not) have done by reading the contents of the Output View.
2. Continue clicking **Step** to execute the next line until you reach the end of the task or until you have determined the issue.

3. You can also run the rest of the task at regular speed from this point forward by clicking **Run from here**or pressing CTRL+SHIFT +F5. Click **Stop** or press CTRL+F2 at any time to stop a step process.

> **NOTE:** If you import a task that includes an unlicensed action/activity, the Steps panel will display the unlicensed step (in **Visual** view) as currently unlicensed. Running a task that includes an unlicensed step will always fail at that step. You can bypass the step by disabling it or removing it altogether.

More on "Stepping"

“Stepping” is the process of executing a task one step at a time. When clicking the **Run** button, task execution may happen so quickly that it’s too complicated to determine what is causing the problem and where that problem takes place. The **Step** button is ideal for such a situation because clicking this button will execute one step, then wait, providing a chance for the developer to examine what the previous step may (or may not) have done or observe the contents generated in the Debug window in order to better diagnose the task. The **Step** button can be accessed by way of the keyboard by hitting the F9 key.

> **NOTE:** Most Task Builder commands and controls contain a corresponding shortcut key (or hot-key). To determine the appropriate shortcut for a control, simply hover the mouse over it.

Control Running Tasks

You can manually pause or stop a currently running task at any time during its execution.

To suspend a running task

- Click **Pause** or press **F6**. Upon doing so, the task delays execution at the current line. Click **Pause** or press **F6** again to resume execution.

To stop a running task

- Click the **Stop** button or press **CTRL + F2**. The task immediately stops at the current line and resets itself.

> **NOTE:** The **Pause** and **Stop** buttons are usually disabled and are active only during task execution.

Task Variables

Task variables - not to be confused with the Variable Action - provide a means of sharing common data within a single function or between more than one function. The parameters of a task variable are similar to that of a task function which comprise a **Name**, **Type**, and **Accessibility** descriptor. In addition, a task variable includes a **Description** parameter as a way to distinguish each variable. Unlike a task function, however, a task variable cannot be marked as **Optional**. Some advantages of task variables are as follows:

- **Preservation of task-centric values** - Data that is common to an entire task or used by one or more functions are candidates for elevation to a task variable. An example could be a variable to indicate whether the task is in “normal” or “error recovery” mode.
- **Can be used as a “named constant”** - In Automate Desktop, a constant is defined at the system level. However, there are situations where a value is meant to be preserved throughout the lifetime of a task, regardless of function or external task, but not dictated by the system configuration. A task variable can accomplish this, optionally with the read-only option set.

- **Allows information-protection and controls access by external tasks** - This can be accomplished by using the Accessibility preference of a task variable. They control which task variables are merged into a parent task or accessible through a task object.

Task variables, functions and events are created and managed through a drop-down UI display that is accessible from the Steps panel of Task Builder .

> **NOTE:** Task Variables are different from Local Variables which are created using the "Variable - Create" action. Task Variables are additional features aimed for advanced users. You can build tasks without the use of Task Variables and may choose to use them at your own pace.

Scope and Accessibility

Scope and Accessibility provide two different capabilities to Automate Desktop and serve two different but complimentary roles necessary to realize the full benefits of task variables. They are primary building blocks that can be used to build a solid foundation for a rich object-oriented approach to tasks.

Scope

Scope limits a task variable’s visibility within a task (that is, using a variable from “the inside”). Without proper scoping, it would not be possible to provide some of the best advantages to function-oriented design, including information-protection, modularity, maintainability and recursion. Scoping helps take large, unruly tasks and enforce logical restrictions to their structure to provide greater readability and maintainability. They can help optimize and enhance task execution speed by reducing the amount of variables that are present in the system at a given time. Scoping also helps avoid unintentional and confusing data changes and variables accesses, leading to easier debugging, and cleaner and more reliable tasks.

> **NOTE:** Task variables are scoped outside of task functions, and all task variables are accessible to all task functions in that task.

Accessibility

Accessibility provides the fundamentals of information-protection, encapsulation and interfacing, all of which are essential for an object-oriented approach to a language. The ability to set task variables and functions as public or private gives a task developer greater control over how a task is used by another task. It makes tasks more portable, manageable and documentable by providing outside access only to those parts of the task that are meant to be used, while providing the user full flexibility of functions. Automate Desktop supports two levels of accessibility:

- **Public** - The variable or function is visible and accessible to external tasks.
- **Private** - The variable or function is not visible or accessible to external tasks.

The accessibility of a task variable can be adjusted to suit the information-protection needs of the task as it relates to an external or sub-task. By default, a task variable is public, meaning it is accessible to an external task or external task function. This is accomplished in different ways, depending on how the external task is referenced, but the behavior of the task variable is identical regardless.

Sub-Tasks

Sub-tasks are task files that are executed within another task by using the Task - Start subtask activity. In this situation, the parent task's (that is, the task executing the Start sub-task step) basic functions, extended functions, public task functions, public task variables, and any local variables (unmarked as private), created up to point where the Start sub-task step is encountered, are accessible to the sub-task. All other variables and functions are not accessible to the sub-task. Conversely, because the Start Task activity is a synchronous operation, the sub-task's public functions and public task variables are not accessible from the parent task. This is fully backward compatible with previous versions of Automate Desktop, since in the past there was only one "function" (what is now called "main"), no task variables, and all (local) variables and extended functions were considered public.

Creating and Defining Task Variables

Task variables are scoped outside of task functions, and all task variables are accessible to all task functions in that task. They are created and presented in a separate panel located on the Steps panel.

To create and define a task variable

1. In the Task Builder's Steps panel, select Task Variables.
2. Click the **Add Task Variable** button. A panel opens where a new task variable can be added and specific parameters can be set.
3. Enter the following parameters:

Property	Type	Description
Name	Text	The unique name of the task variable to create.
Type	Text (options)	The type of task variable to create. The available options are: • **Variable** - The return value will be in the form of a variable. • **Array** - The return value will be in the form of an array. • **Dataset** - The return value will be in the form of a dataset.
Access Type	Text (options)	The task variable's access type. The available options are: • **Public** - The variable is visible and accessible to external tasks. • **Private** - The variable is not visible or accessible to external tasks.
Initial Value	Text	The initial value of the task variable (if any).

Property	Type	Description
Description	Text	An optional description of the task variable.

4. When finished, click **Save Changes**. The newly created task variable along with its properties appears on the list of task variables available for the task.

Editing and Deleting Task Variables

To edit an existing task variable

1. From the task variables panel, select the variable you wish to edit from the list and then click **Edit**.
2. Make the desired modifications then click **OK** when finished.
3. Reflected changes appear in the main panel.

To delete an existing task variable

1. From the task variables panel, select the variable you wish to delete and click the **Remove** button.
2. The task variable is permanently removed from the list of available task variables.

Creating Variables (Advanced Workflow Module)

A variable (also known as a local variable) is a placeholder for varying or changeable data. Variables play an important role in because they enable developers to write flexible tasks. Rather than entering data directly into a task step, a developer can use variables to represent the data. Then, when the task runs, the variables are replaced with real data. This makes it possible for a task to perform actions on values that may or may not be known until runtime. It also allows a single task the ability to hold different sets of data.

Variables are created and/or set in Task Builder during task creation. They are commonly used when a task involves collecting data from a source and then performing some action on it. You use the variable to contain the collected data, and then set up the actions to be performed on the data by referencing the variable. For example you could create a variable in the beginning of a task that will be populated with the user's input in a form or message box. In a subsequent step you could perform calculations on the data the user entered by referencing the variable.

Creating Variables

The Create variable activity generates a local variable which can be used to store dynamic values for utilization in any step of the task or any sub-tasks started with the Start Task action. The developer can enter a value to initially populate the variable during creation or the value field can be left blank. Instead, variables can be set with a value using the Set Variable action, which adds or changes the contents of an already existing variable. Certain available actions that support populating variables can also set or modify a variable's contents, such as the Input Box action. This action displays an input box allowing the user to enter a value which is saved to the variable specified.

Once a variable is created, it becomes available for use in subsequent steps of the task. It will appear on drop-down lists in places where a variable can be entered, and it can be used in expressions by simply placing the name of the variable between percent signs (%). For example, entering %UserInput% tells to populate the expression with the current value of the variable named UserInput.

It is important to note that in order for a variable to be used in a task step, it must initially be created within an earlier step.

Variable Naming Conventions

Variable names must contain only alphanumeric characters, must start with a letter and cannot contain spaces. Variable names are not case-sensitive. When choosing a variable name, it is good practice to select a name that is descriptive of what the variable holds. For example, if a variable holds the size of a shoe, then name it ShoeSize or TheSize. This makes the task more comprehensible. Also, be sure to avoid using BASIC keywords, functions, or instructions. Names such as DATE or TIME would create a conflict. One way to

avoid this is to include distinguishing characters in the variable name such as VAR, THE or MY. For example, a date variable could be named MyDate, DateVar or theDate.

Variable names must be unique within a task, but can be repeated from one task to the next. For example, if you create a variable in one task named UserInput, you can create a variable of the same name in another task without any conflict occurring.

Variable naming restrictions

There are limitations to the names that can be given to variables. These limitations derive from the fact that when the expressions between % signs are evaluated as a script, even if they are a simple variable name. Therefore all BASIC Script keywords, functions names, and operators are forbidden as variables names (e.g. while, wait, like, instr, date, time, now, daysinyear, etc).

Also all Windows environment variables such as PATH, DATE, TMP, PROMPT are also reserved and cannot be declared as new variables. They are, however, accessible and readable, whether or not they were declared “as parameters” in an AMVARIABLE statement.

This is quite a lot of common names which are in fact reserved. Nothing warns that they are not allowed, but the error will appear at execution time, with some strange error message that is all but obvious to understand. A wise precaution to avoid any mistake is to initiate all variable names by a specific (group of) character(s), say v_ (underscore) which then allows variables like v_wait, v_do, v_while, v_path without any ambiguity.

Scoping & Accessibility

Local variables created within a task function (e.g., by using Create Variable, Create Array, etc.) are scoped to that function. Local variables, therefore, are not visible and thus cannot be used outside the function in which they are created. If a variable is created within a function with the same name as a task variable, the local variable “hides” the task variable and takes precedence.

Scoping a created variable to the function in which it is created has the following advantages:

- Variables are only created when required - Because functions are distinct units designed for a single purpose, variables created within them are typically only needed to serve that purpose before returning an end result. Scoping the variables to the function keeps these variables localized to where they are needed. Once their purpose is complete, they are discarded.
- Keeps temporary variables where they are needed - Many functions make use of temporary variables, for example when indexing an array, looping, etc. Local variables provide a means to create indexing or temporary variables that are only available within that function without creating a global variable that can be trampled on by other functions. An example of the danger of using a common index variable without local scoping is creating an index variable named i that, in a loop, calls another function which itself uses a variable i as an index. When the second function is finished, the first function will have the wrong value! Local variables avoid this problem.
- It does not preclude the use of task variables - A function can still set and retrieve the value of a task variable, or assign a task variable the value of a local variable. This improves readability and keeps the number of variables a user needs to remember or debug to a minimum.
- Makes tasks easier to debug - The use of local variables and task variables means there are less variables the user needs to keep in their head at any given time. “The less you have to keep in mind, the smaller the chance that you’ll make an error because you forgot one of the many details you needed to remember.” Additionally, it provides a means to separate variables in a debugger or other visual display.
- Allows recursion - Recursion is impossible without local variables. Recursion relies on local variables to store the state of the function during each call. If a variable is scoped outside the function, each call of that function will overwrite the data of the previous call, thus rendering the recursion useless.

A local variable’s or local array's accessibility is defined by the Variable is private option on the Create Variable or Create Array step. By default, this option is set to OFF and therefore

the variable's default accessibility is public. This means that an external task can access a public local variable.

Sub-tasks

Sub-tasks are task files run synchronously at the step level using the Start Task action. Sub-tasks are treated as individual modules that maintain their own scope and accessibility. Sub-tasks can access public variables, functions and extended functions of the parent task, but not vice versa. Functions and variables contained in the sub-task, however, whether private or public, are accessible to functions called within the sub-task while the sub-task is executing, exactly as if the sub-task were running independently. An important point to note regarding this is how events (which are functions that are optionally implemented by a task and called implicitly by the task engine) are scoped; a sub-task's events are fired during sub-task execution, not the parent's. This follows the rules of variable and function scoping and accessibility.

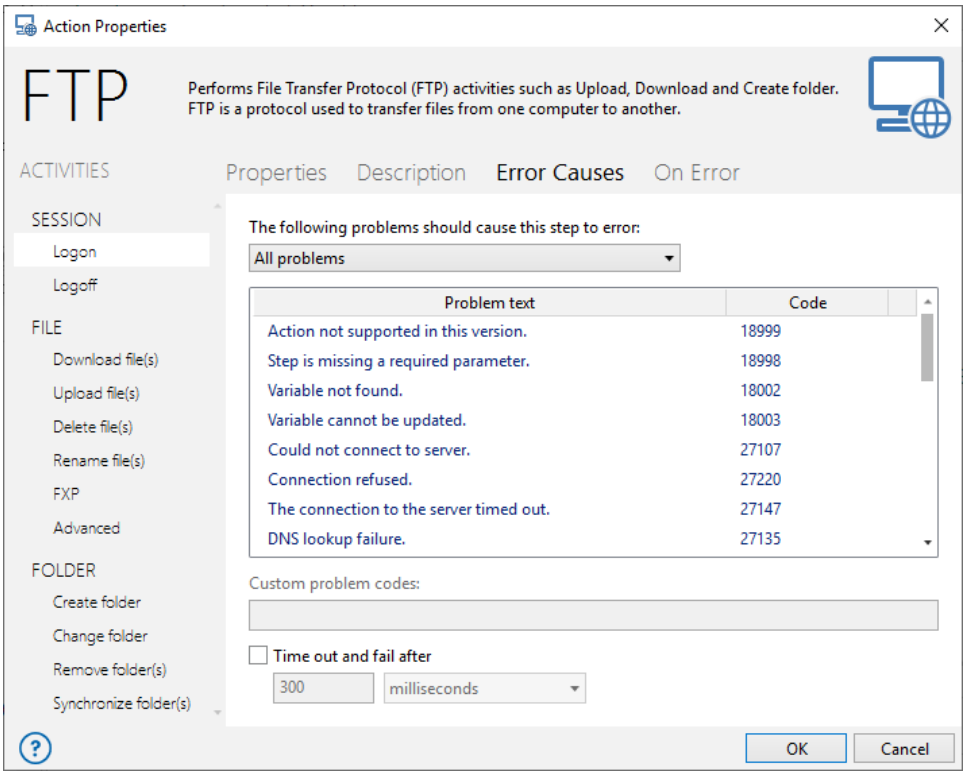
Error Causes

The **Error Causes** properties allows you to modify how this step should behave upon the occurrence of an error. The default behavior is to monitor for the occurrence of all errors, however, these properties can be set in order for this step to ignore certain errors or only react to specific errors. It also allows you to set the length of time this step should be permitted to execute before causing a time out error. If any **Error Causes** conditions are met during task execution, the procedures set under the On Error properties are carried out.

> **NOTE:** All **Error Causes** properties are optional entries.

Practical usage

Particularly used as a way to ignore non-critical errors or overlook anticipated issues in order for task execution to proceed without interruption. Can also be used to set a time out value for time-sensitive steps.


Action Properties
FTP
Performs File Transfer Protocol (FTP) activities such as Upload, Download and Create folder.
FTP is a protocol used to transfer files from one computer to another.
ACTIVITIES
Properties
Description
Error Causes
On Error
SESSION
Logon
Logoff
FILE
Download file(s)
Upload file(s)
Delete file(s)
Rename file(s)
FXP
Advanced
FOLDER
Create folder
Change folder
Remove folder(s)
Synchronize folder(s)
The following problems should cause this step to error:
All problems
Problem text
Code
Action not supported in this version. 18999
Step is missing a required parameter. 18998
Variable not found. 18002
Variable cannot be updated. 18003
Could not connect to server. 27107
Connection refused. 27220
The connection to the server timed out. 27147
DNS lookup failure. 27135
Custom problem codes:
Time out and fail after
300
milliseconds
OK
Cancel

Parameters				
Property	**Type**	**Default**	**Markup**	**Description**
The following problems should cause this step to error	Text	All Problems	`AM_ERRORS=18999,18998,20008`	Allows you to select / deselect specific errors that will cause this step to fail. The available options are: • **All Problems** - Any problem encountered during execution of this step will cause the instructions specified in the **On Error** properties to be carried out. • **Selected Problems** - Only the selected errors will

Parameters				
Property	**Type**	**Default**	**Markup**	**Description**
				cause the instructions specified in the **On Error** properties to be carried out. Non-selected errors are ignored. • **All except Selected** - All non-selected errors will cause the instructions specified in the On Error properties to be carried out. Selected errors are ignored.

Parameters				
Property	**Type**	**Default**	**Markup**	**Description**
Custom problem codes	Text	(Empty)	`AM_ ERRORS="18999,18998,20008"`	Allows a custom error code to be entered in case it does not appear in the list of errors. This parameter will adhere to the specifications set under the parameter labeled **The following problems should cause this step to error.** Variables can be used in this parameter. **NOTE:** Use AMError to determine specific information about an error.

Parameters				
Property	**Type**	**Default**	**Markup**	**Description**
Time out and fail after	Number	(Empty)	`AM_TIMEOUT=20`	If enabled, allows you to set the total amount of time this step should be allowed to execute before generating a timeout error. For example, a SQL Query step is set to time out after 30 seconds. If the query took longer than 30 seconds to complete, the step would generate a time out error and the instructions set in the **On Error** properties would be carried out. This parameter is disabled by default.

Parameters				
Property	**Type**	**Default**	**Markup**	**Description**
Timeout scale	Text (options)	Milliseconds	`AM_TIMEOUTSCALE=seconds`	The scale in which the timeout value should be set to. The available options are: • **Milliseconds (default)** - The timeout value is measured in milliseconds. • **Seconds** - The timeout value is measured in seconds. • **Minutes** - The timeout value is measured in minutes. • **Hours** - The

Parameters				
Property	**Type**	**Default**	**Markup**	**Description**
				timeout value is measured in hours.

On Error

The **Error Causes** properties allows you to modify how this step should behave upon the occurrence of an error. The default behavior is to monitor for the occurrence of all errors, however, these properties can be set in order for this step to ignore certain errors or only react to specific errors. It also allows you to set the length of time this step should be permitted to execute before causing a time out error. If any **Error Causes** conditions are met during task execution, the procedures set under the On Error properties are carried out.

> **NOTE:** All **Error Causes** properties are optional entries.

Practical Usage

Particularly used as a way to ignore non-critical errors or overlook anticipated issues for task execution to proceed without interruption. Can also be used to set a time out value for time-sensitive steps.

Parameters				
Property	**Type**	**Default**	**Markup**	**Description**
The following problems should cause this step to error	Text	All Problems	`AM_ERRORS=18999,18998,20008`	Allows you to select / de-select specific errors that will cause this step to fail.The available options are: • **All Problems** - Any problem encountered during execution of this step will cause the instructions specified in the **On Error** properties to be carried out. • **Selected Problems** - Only the selected errors will

Parameters				
Property	**Type**	**Default**	**Markup**	**Description**
				cause the instructions specified in the **On Error** properties to be carried out. Non-selected errors are ignored. • **All except Selected** - All non-selected errors will cause the instructions specified in the On Error properties to be carried out. Selected errors are ignored.

Parameters				
Property	**Type**	**Default**	**Markup**	**Description**
Custom problem codes	Text	(Empty)	`AM_ ERRORS="18999,18998,20008"`	Allows a custom error code to be entered in case it does not appear in the list of errors. This parameter will adhere to the specifications set under the parameter labeled **The following problems should cause this step to error.** Variables can be used in this parameter. **NOTE:** Use AMError to determine specific information about an error.

Parameters				
Property	**Type**	**Default**	**Markup**	**Description**
Time out and fail after	Number	(Empty)	`AM_TIMEOUT=20`	If enabled, allows you to set the total amount of time this step should be allowed to execute before generating a timeout error. For example, a SQL Query step is set to time out after 30 seconds. If the query took longer than 30 seconds to complete, the step would generate a time out error and the instructions set in the **On Error** properties would be carried out. This parameter is disabled by default.

Parameters				
Property	**Type**	**Default**	**Markup**	**Description**
Timeout scale	Text (options)	Milliseconds	`AM_TIMEOUTSCALE=seconds`	The scale in which the timeout value should be set to. The available options are: • **Milliseconds (default)** - The timeout value is measured in milliseconds. • **Seconds** - The timeout value is measured in seconds. • **Minutes** - The timeout value is measured in minutes. • **Hours** - The

Parameters				
Property	**Type**	**Default**	**Markup**	**Description**
				timeout value is measured in hours.

Error Handling

The Task Builder's three-tiered approach to error handling provides a variety of methods in which to troubleshoot problematic tasks as well as supply alternative techniques to allow a task to recover gracefully from unexpected errors. Error handling can be set by way of the step, task and system level. The flow of error handling takes precedence from step to task to system. For instance, if an error occurs during task execution, the Task Builder first adheres to any exception handling parameters set for the step that caused the error (if any), followed by error handling set for the task level, and conclusively, any system wide error handling parameters.

> **NOTE:** In the Automate Desktop that is bundled in EFT, the Managed Tasks in the Task Administrator are not triggered by EFT event rules. Instead of the Automate Task Administrator, events and triggers are managed by the EFT Event Rules system. (You can copy the managed tasks' AML file (**\ProgramData\Automate\Automate Desktop 2024\Tasks**) and then import or copy it to a task in EFT.)

Step-Level Error Handling

Step-level error handling handles task exceptions on a per step basis. Normally, any failure of a step within a task causes a step error. What causes a step to fail is, however, dependent on the step and it's parameters. These parameters can be adjusted under the **Error Causes** tab of any Task Builder step (as shown below). For additional details, see Error Causes.

If a step fails, the user can select from a number of actions to be carried out. These actions are located in the **On Error** tab of any specific step (as shown below).

Step level error handling takes precedence over task and global lever error handling. When a step level error occurs, a task can alert a user, request user intervention, or with proper exception handling, it can recover on its own.

Task-Level Error Handling

As part of The Task Builder's "tri-level" error handling, errors may be handled on a per task basis. This is ideal if the user wants the Task Builder to perform specific exception handling procedures for a managed task. A task level error occurs when a step within a task fails and Task Builder generates an error.

In most cases, error information is written to a log file so the user can easily track down illusive errors or identify obscure program behavior.

System-Level Error Handling

The Task Builder can handle errors globally for all tasks that run on a system. If any system level error handling parameters are enabled, the Task Builder will act on ALL managed tasks that fail with an error.

Debug Panel Overview

The Debug panel contains nine debug views separated by tabs (the active view's tab is always colored blue). Here, you can select the **Variables** tab to easily view variables and datasets that are contained in the task as well as their initial and current values, or click the **Output** tab to view step by step information about the task during execution. Each debug view supplies a context menu containing relevant options and commands for that particular view.

Debug Panel Views

The available debug tools are listed below:

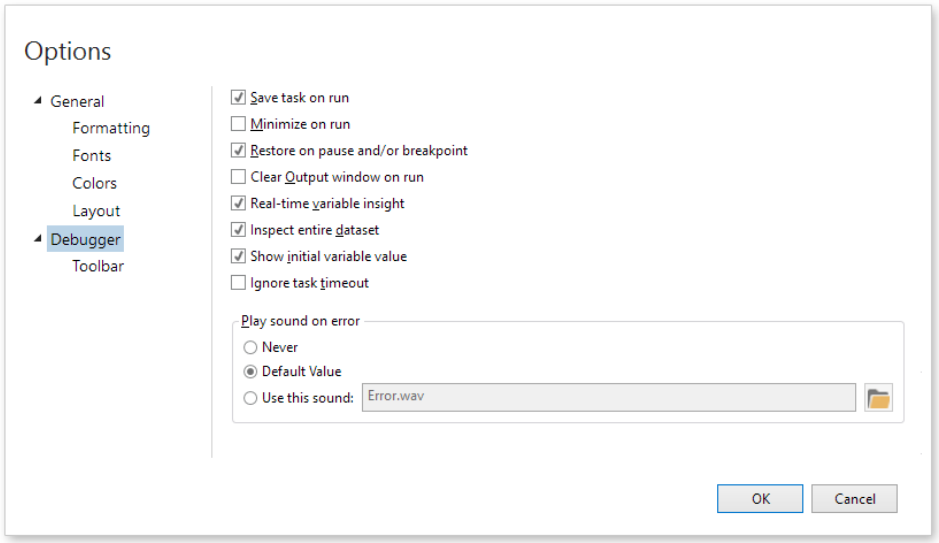
Debug Panel View	Description
Debug Panel - Attachments	Lists all the files that are attached to the current task.
Debug Panel - Breakpoints	Lists all breakpoints contained in the task and their corresponding steps.
Debug Panel - Comments	Displays each comment in the task along with its step number.
Debug Panel - Labels	Lists all labels in the task, including the label name and corresponding step number.
Debug Panel - Output	Shows detailed step by step information about the task during execution.
Debug Panel - Regions	Displays the list of regions contained in the task, including region name and corresponding step number.
Debug Panel - Sessions	Displays a list of steps that contain sessions within the task.
Debug Panel - Stack	Displays the depth of the currently running task. When a sub-task is running, its immediate parent is listed below it.
Debug Panel - Variables	Lists variables and datasets that are contained in the task as well as their initial and current values.
Debug Panel - Watches	Shows all currently set watches and displays the evaluated values for the watched variables and expressions as the task runs.

Task Builder Debugger Options

The Task Builder includes settings that affect its behavior while running a task. In some cases you may want to change these settings to better fit your debugging needs. For instance, you can opt to minimize the Task Builder window during execution to better examine an interactive task, automatically clear the Output Debug Panel after each run, or play a custom sound when an error occurs.

To access Task Builder's Debugger Options page

- Click to **File> Options>Debugger**. Debugger options is a subset of Task Builder Options.



Parameters

The following table describes available Debugger options:

Property	Default	Description
Save task on run	Disabled	If enabled, Task Builder automatically saves changes made to a task when clicking the **Run** button. If the task has not been saved yet, Automate Desktop will prompt the user whether or not to save before running. If disabled, changes will not be saved upon run.
Minimize on run	Disabled	If enabled, Task Builder minimizes to the system tray when execution starts. When the task ends, Task Builder restores itself onto the desktop and highlights the current step or displays **Finished** as the step status in the status bar. If disabled, Task Builder stays on the desktop (does not minimize or hide itself) during task execution. > **NOTE:** It is recommended to enable this option when testing tasks that include interactive steps (for example, Send Keystrokes, Move Mouse to Object actions) or steps that interact with other windows (for example, Maximize Window, Minimize Window actions). This is due to leaving the Task Builder window on the desktop may interfere with other windows that appear during runtime. Doing so may result in a Send Keystrokes step, for example, failing to send keystrokes to the proper window or application.
Restore on pause and/or breakpoint	Enabled	If enabled, when task execution is manually paused (when a user clicks the **Pause** button) or pauses at a breakpoint, and the Task Builder window is currently minimized, it restores itself and highlights the current step. When execution continues, the Task Builder window, once again, minimizes to the system tray.

Property	Default	Description
Clear Output window on run	Enabled	If enabled, clears the contents of the Output window before each task execution . If disabled, new debugging output will append existing output accumulated from previous runs.
Limit entries in output panel to	Disabled	If enabled, limits the number of entries in the Output panel to the numeric value entered here. Once the value is exceeded, the oldest entry is deleted. If disabled, entries are not limited.
Real-time variable insight	Enabled	If enabled, the Variables window is updated after each step to show the current value of existing variables on a step by step basis. If disabled, the Variables window is updated only when the task ends or when it is in a paused state (by clicking the **Pause** button, when a Breakpoint is encountered or while stepping through with the use of the **Step** button). **WARNING:** Enabling this option could cause Task Builder to run tasks slower.
Inspect entire dataset	Disabled	If enabled, during runtime, extended dataset properties that are normally hidden (for example, creation date/time, total rows) will be displayed in the Variable window. If disabled, only basic dataset properties will be displayed. **WARNING:** Enabling this option could cause Task Builder to run tasks slower.
Show initial variable value	Enabled	If enabled, causes the Variables window to include a column for **Initial Value** in addition to the **Current Value** column. If disabled, only the current value is shown.

Property	Default	Description
Play sound on error	Error.wav	Denotes whether to play a sound when the task stops as a result of an error. The selections are: • **Never** - Do not play a sound when an error occurs. • **Use default sound** - Use the default system error sound, as specified in Window's Control Panel. • **Use this sound** - Specify a valid path and file name of a .wav file to use.

Debugging Tools

Task Builder allows you to visually construct and troubleshoot the steps that an Automate Desktop task performs before it is put into production. It contains many debugging tools that makes it faster and easier to find and resolve issues in your task. You can use breakpoints to pause execution and observe certain elements of a specific step in order to pinpoint the cause of an errant task, or use the various run options to navigate through the steps of your task in different ways, including the ability to start execution from a specific step, run only selected steps, or run each step one by one. The Debug panel allows you to view variable and dataset values or evaluate expressions while the task is suspended.

Debugging Tools and Techniques

The table below describes various Task Builder debugging techniques that you can apply in order to test and troubleshoot a task:

Technique	Description
Attachments	Jumps to a particular step in the Steps panel where a breakpoint resides.
Bookmarks	Removes the selected breakpoints. To select multiple breakpoints, hold down CTRL during selection.
Breakpoints	Removes all breakpoints contained in the task.
Comments	Opens the help topic regarding this subject.

Technique	Description
Regions	Jumps to a particular step in the Steps panel where a breakpoint resides.
Run Options	Removes the selected breakpoints. To select multiple breakpoints, hold down CTRL during selection.
Watches	Removes all breakpoints contained in the task.

> **NOTE:** If you import a task that includes an unlicensed action, the Steps panel will display the unlicensed step (in **Visual** view) as currently unlicensed. Running a task that includes an unlicensed step will always fail at that step. You can bypass the step by disabling it or removing it altogether.

Debug Panel - Output

The Output debug panel displays detailed information about a running task and usually provides the best indication of why a task failed. As a task executes, information about each step is logged to the Output panel in real-time, including task step, the action being performed within the step and completion or failure status. If the task fails, extended error information is logged in red for easy detection.

The Output panel is a useful debugging tool because it can contain invaluable information about what a task is doing at any given time. Furthermore, since the data being output is specific to the action being performed, pertinent information unique to each activity's execution may be logged to the Output panel. For example, the File System - Copy activity outputs the name of each file being copied while it is executing. You can determine if the proper files are being copied by simply viewing the Output panel.

Parameters

The Output debug window includes the following fields (columns):

Column header	Description
Step	The step number that the output message refers to.
Function	The function being executed.
Message	A description about the current execution.

Context Menu

Right-clicking inside the Output panel opens a context menu that contains the following items:

Menu item	Description
Auto Scroll	If enabled, causes the Output panel to automatically scroll down to the latest output message that appears during runtime. If disabled, auto scrolling will not be applied.
Copy	Copies the selected (currently highlighted) output data to the clipboard. The contents can then be pasted into any application or window that accepts standard text.
Select All	Selects all data currently displayed in the Output panel.
Clear	Clears all data currently displayed in the Output panel.
Save As...	Saves output data as a (.log) file.
Help	Opens the help topic regarding this debug tool.

Managing Output Data

The contents of the Output panel can be handled in several different ways. New output can be appended to the existing data in the **Output** window, or the window can be cleared before additional data is entered. Output data can be copied to the clipboard, and the contents can then be pasted into any application or window that accepts standard text. You can also save the contents of the **Output** tab to a standard text file.

To copy the contents of the Output panel

1. Right-click anywhere in the **Output** panel, and then select **Copy**.
2. Open the application of your choice (usually a text file) and use the **Paste** function.

To clear the contents of the Output window

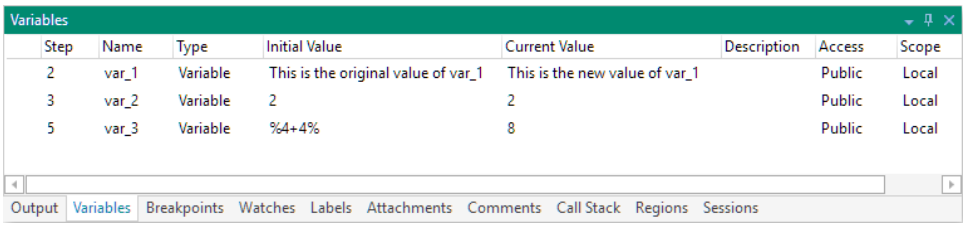
- Right-click anywhere in the **Output** window, and then select **Clear**.

To save output data to a text file

1. Right-click anywhere in the **Output** window, and then select **Save As** .
2. Select a file location (usually a text file).
3. Enter a file name, and then select **Save**.

Debug Panel - Variables

The Variables debug panel is ideal in debugging object types that may contain dynamic data, such as variables, arrays, JSON Objects, and datasets. When a task runs, this panel generates real-time information about such objects including their initial and current value. You can examine content without inserting additional steps to output the values. Similarly, you can insert breakpoints at certain points in your task to halt execution and view this panel to determine if the current values have been properly set or modified.



Parameters

The Variables panel generates information into these columns:

Column header	Description
Inspect	Displays the Inspect icon when a step in the panel is selected. Clicking the icon opens the corresponding dialog, allowing you to further inspect the selected object type.
+/-	When an parent object type has child objects within it, click the expand/collapse icon to expand the parent to view the children. Click the icon again to collapse the parent object type.
Step	The step number in the Steps panel associated to the object type.
Name	The name of the selected object type.
Type	The type of data being displayed, such as a variable, array, JSON Object, dataset, or field (associated to a dataset).
Initial Value	Before the task is run for the first time in a Task Builder session, this column contains the initial value of the variable as set in the variable's/array's properties. If no value was set, this column is blank. An array or dataset can contain multiple rows and columns. During runtime, this section will list the initial value of the rows and columns associated to each array/dataset.
Current Value	This column reflects the current value of the object type during task execution. **NOTE:** If **Real-Time Variable Insight** is enabled in the Debugger Preferences, the value is updated after each step executes. If **Real-Time Variable Insight** is not enabled, the variable **Current Value** field is updated only when the task is paused, when a breakpoint is encountered or when the task ends.

Column header	Description
Description	A description of the variable or array as set in the associated Variable - Create or Array - Create properties dialog. The description does not affect variable/array performance at runtime.
Access	Whether the object type is public or private. Controls how variables are accessed externally.
Scope	Specifies whether the object type's scope is Local or Task: • **Local** - Specifies that the object type is local to the current context or scope. Usually, this means the procedure or function you are currently executing. • **Task** - Specifies that the object type is a task variable which is considered global to the entire task.

Context Menu

Right-click anywhere inside the Variables panel to view the following menu items.

Item	Description
Properties	Opens the properties dialog of the selected object type for viewing or editing purposes.
Inspect	Opens a dialog allowing you to further inspect the selected object type.
Jump to	Jumps to the step number in the Steps panel associated to the object type.
Sort	Sorts the list of object types in the following order: • **Declaration (default)** - Sorts by order of declaration. • **Ascending** - Sorts in ascending alphabetical order. • **Descending** - Sorts in descending alphabetical order.
Copy Name	Copies the name of the object type to the clipboard.

Item	Description
Copy Value	Copies the value of the object type to the clipboard.
Add Variable	Adds a variable to the current task (identical to selecting the Variable - Create activity from the Actions panel).
Add Array	Adds an array to the current task (identical to selecting the Array - Create activity from the Actions panel).
Add Dataset	Adds a dataset tot he current task (identical to selecting the Dataset - Create activity from the Actions panel).
Add to WatchList	Adds the selected object type to the Watches debug panel to examine the state of it while a task is running.
Delete	Deletes the selected object type from the Variables panel.
Delete Locals	Deletes all local object types in the Variables panel.
Delete All	Deletes all objects listed in the Variables panel.
Help	Opens the help topic regarding this debug tool.

Managing Object Types

Object types can be added, edited, or deleted directly from the Variables debug panel. When you edit variables in this way, steps are added, changed, or deleted in the task.

To add a variable, array or dataset object type

1. Right-click anywhere inside the Variables debug panel, and then select **Add Variable**, **Add Array**, or **Add Dataset** from the context menu that appears.
2. Enter the desired properties in the properties dialog, and then click **OK** when finished.

To view/edit an existing object type

1. In the Variables debug panel, double-click on the object type, or right-click on it and select **Properties** from the context menu that appears.
2. View/edit the desired properties in the dialog, and then click **OK** when finished.

To delete an object type

- In the Variables debug panel, right-click on the object type, and then select **Delete**, or select the object type, and then press the **Delete** key on your keyboard.
- To delete only local object types in the Variables debug panel, right-click anywhere inside of it, and then select **Delete Locals**.
- To delete only local object types in the Variables debug panel, right-click anywhere inside of it, and then select **Delete All**.

Debug Panel - Breakpoint

Breakpoints provide a means of pausing execution at a specified step in order for you to inspect certain aspects of the task. In the Steps panel, breakpoints are designated by a red, circled icon located on the left side of the step number. The Breakpoints debug window is used for viewing and managing breakpoints. It lists the step number and textual description of any step that contains a breakpoint. Using this debug window, you can easily jump to any breakpoint step, remove a specific breakpoint or remove all breakpoints. More on Breakpoints

Parameters

The Breakpoints debug screen includes the following fields (columns):

Column header	Description
Step	The step number where the breakpoint resides.
Description	A textual description of the activity where the breakpoint resides. This is identical to the visual step description displayed in the Steps panel.

Context Menu

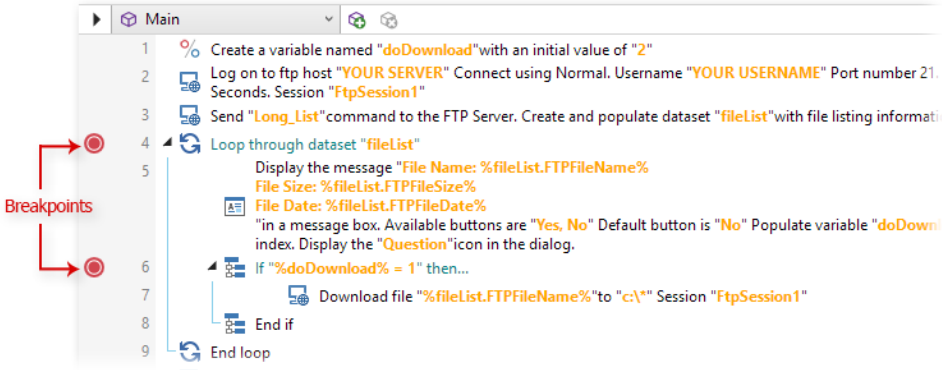
Right-clicking inside the Breakpoints screen opens a context menu with the following items:

Menu Item	Description
Jump To	Jumps to a particular step in the Steps panel where a breakpoint resides.
Remove	Removes the selected breakpoints. To select multiple breakpoints, hold down CTRL during selection.
Remove All	Removes all breakpoints contained in the task.
Help	Opens the help topic regarding this debug tool.

Setting Breakpoints

When a task runs, execution time of each step may only amount to 1 or 2 seconds, making it difficult to examine certain steps that may not be running properly. Breakpoints may be the most ideal solution in this situation. They enable you to suspend task execution where and when you need to. Entering break mode does not stop or end the execution of your task. Elements, such as variables, datasets and functions remain in memory, but their movements and activities are suspended. During the interruption, you can view the Debug panel to acquire knowledge about your task and determine if it is functioning as expected.

Breakpoints can be set at any step and can be used as many times as needed in a single task. They take effect only when a task is run from Task Builder and are ignored when the task is triggered or executed manually outside of Task Builder. Breakpoints are designated by a red, circled icon located on the left side of the step number in the Steps panel. The Breakpoints debug window can be used to examine and manage breakpoints within a given task. Using breakpoints can speed up the debugging process enormously. Without this feature, it would be very difficult to debug a large task.



General Behavior

When a running task reaches a breakpoint step, Task Builder pauses execution, at which point, you can verify proper completion of a particular step or examine the data produced by the assortment of debugging windows located in the Debug panel. This is made possible because a paused task still remains in a running state but suspended between actions. Variables and datasets retain their values and the Output window continues to display detailed information about the task up to the suspended step. You can view step

details generated by the Output window or view the Variables window to determine the current value of a certain variable or dataset. In addition, you can manage existing breakpoints or easily jump to a breakpoint step by way of the Breakpoints window.

Depending on what you determine from the data displayed, you can then click **Continue** from the ribbon to continue the task, click **Step** to continue the task step by step, or click **Stop** to halt task execution and reset all debugging elements.

> **NOTE:** Breakpoint symbols are positioned in the same location as bookmark symbols on the Steps panel. If a specific step comprises both, the icon will contain a bookmark symbol overlaying a breakpoint symbol.

To set a single breakpoint, do one of the following

- From within the Steps panel, click the gray margin to the left of the step number on the line where you want to set the breakpoint.
- Select (highlight) the step you want to place a breakpoint and on the **Home** tab of the ribbon, click the **Breakpoint** command from the **Step** command group.

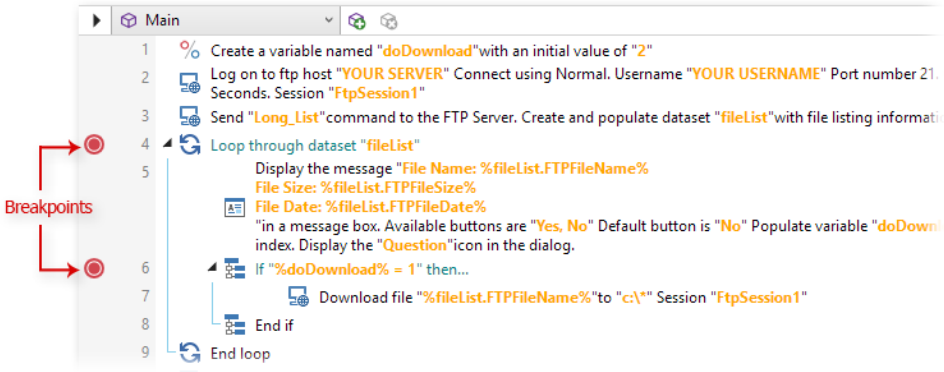
To set multiple breakpoints simultaneously

1. From the Steps panel, select the steps in which to place a breakpoint. To select more than one step, hold down CTRL during selection.
2. From the Ribbon control, go to the **Home** tab and click **Breakpoint** from the **Step** command group.

About Bookmarks

Bookmarks are a means of marking steps in a task with placeholders so you can find them easily during construction or testing. Bookmarks can be set at any step and can be used as many times as needed in a single task. They take effect only when a task is run from Task Builder and are ignored when the task is triggered or executed manually outside of Task Builder. In the Steps panel, bookmarks are indicated by a blue ribbon icon located on the left side of the step number (as shown below). You can mark steps in your task with bookmarks

and then navigate directly to those steps by clicking the **Next Bookmark** or **Previous Bookmark** ribbon controls.

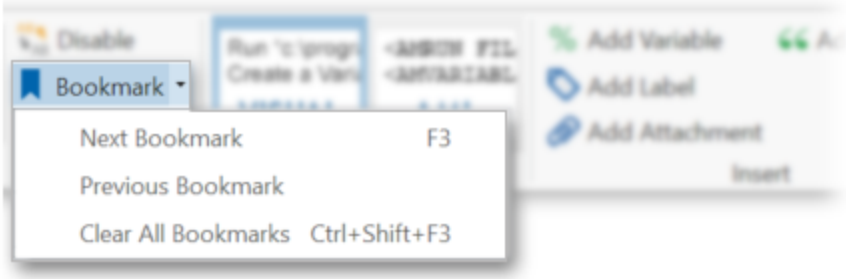


To set a bookmark

1. In the Task Builder's Steps panel, select the steps to bookmark. To select more than one step, hold down CTRL during selection.
2. From the ribbon's **Home** tab, click **Bookmark** from the **Step** command group.

To navigate to a bookmark

1. From the ribbon control's **Home** tab, click the down arrow located on the right side of the **Bookmark** control.
2. Click **Next Bookmark** to jump to the next bookmark or **Previous Bookmark** to jump to the previous bookmark.



To clear a bookmark

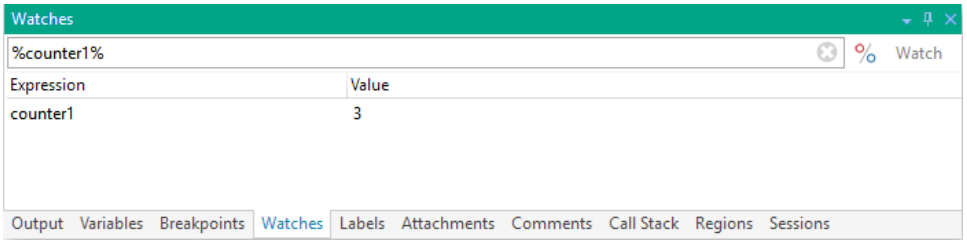
1. In the Task Builder's Steps panel, select the steps that contain the bookmarks you wish to clear. To select more than one step, hold down CTRL during selection.

2. From the ribbon's **Home** tab, click **Bookmark** from the **Step** command group.

> **NOTE:** Bookmark symbols are positioned in the same location as breakpoint symbols on the Steps panel. If a specific step comprises both, the icon will display a bookmark symbol overlaying a breakpoint symbol.

Debug Panel - Watches

The Watches debug panel can store several variables and expressions that you want to view over the course of the debugging session. You can set a watch on a variable used in the task or you can enter an expression to create a more complex watch. You can use the Watches debug view to add, remove, or change watches. After each step executes, any variables and expressions listed are re-evaluated and updated with their current values. Watches go beyond the Variable window by providing an intuitive means of testing variable or task states while a task is executing. More on Watches



Parameters

The Watches window includes the following fields (columns):

Column header	Description
Expression	The syntax of the watch. This can be simply a variable name or a complex expression.
Value	The result of the watch expression. This is updated with each step execution while the task is running.

Context Menu

Right-click inside the Watches window to view the following menu items:

Menu item	Description
Change	Renames the selected variable or modifies the selected expression.
Remove	Removes the selected item. To select multiple items, hold down CTRL during selection.
Remove All	Removes all items listed.
Help	Opens the help topic regarding this debug tool.

Using Watches

A Watch is a debugging feature used for examining the state of variables and other expressions within a running task. A watch can simply be the name of a variable. For example, to watch the value of a variable named `TotalCount`, add a watch that contains simply `%TotalCount%`. Embedded expressions can also be used within a watch, using the same syntax as a standard Automate Desktop embedded expression. For example, to watch the length of an Automate Desktop variable named myName, add a watch `%Len(myName)%`.

The Watches debug panel allows you to add, remove or modify items to watch. It displays the list of added watches and updates their value as the task runs. This way, watches can be re-evaluated after each task step is executed. Watches are mainly used for debugging, therefore, they are ignored when a task is executed as a result of a trigger or executed manually from the Task Administrator.

The Watches debug panel is a place where you can enter variable names and expressions that you want to watch during a debugging session.

To watch a variable

- Enter the variable to watch in one of the following ways:
    - In the Steps panel of the Task Builder, right-click the Create variable step associated with the variable you want to watch and select **Add to WatchList**.

- In the Variables debug panel of the Task Builder, right-click the variable you want to watch and select **Add to WatchList**.
- In the Watches debug panel, type the name of the variable in the provided text box and click **Watch**.

To watch an expression

1. In the Watches debug panel, enter the expression to watch in one of the following ways:
   a. Type the expression into the text box. For example, to evaluate the length of an Automate Desktop variable named `myName`, enter `Len(myName)`.
   b. Click the percent icon located to the right of the Watches text box. When the Expression Builder dialog appears, use it to create an expression to watch. Click **Insert** when finished.
2. Click **Watch**.The expression and its current value (if applicable) is added to the **Watches** debug panel.

To modify a watch

1. In the **Watches** debug panel, right-click the desired watch and select **Change**.
2. Enter the new variable or expression name for the watch, and then press **Enter**.

To remove a watch

- Select the watch to be removed, and then press the **Delete** key.
- Right-click on the desired watch, and then select **Remove** .
- To remove all watches, right-click on any watch and select **Remove All**.

> **NOTE:** Using watches can affect runtime performance in the debugger on slower machines. If debugging speed becomes a factor, try removing watches that are no longer needed. This does not apply to runtime performance outside the debugger, as watches are ignored when the task is executed outside Task Builder

Debug Panel - Labels

A Label can be used as a "bookmark" in a task. Labels can be placed at specific steps in a task to mark them as reference points for Goto activities. Labels can be used in step level error handling by setting up a Goto action to be executed if a step error occurs. The Labels debug panel displays labels that are contained in a task, including the name of the label and corresponding step number. This debug tool also allows you to create labels to add to the task. More on Labels

Parameters

The Labels debug screen includes the following fields (columns):

Column header	Description
Step	The step number where the label resides.
Label	The name of the label (this can be a brief description of the label).

Context Menu

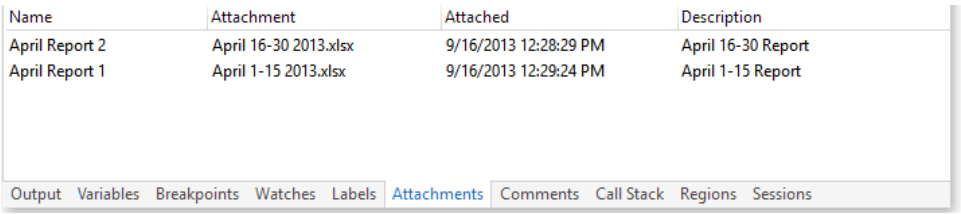
Right-clicking anywhere inside the Labels screen opens a context menu with the following items:

Menu Item	Description
Add	Adds a label to the task. To add a label to a specific step, highlight the step from the Steps panel, then right-click inside the **Labels** tab and select **Add**.
Properties	Opens the properties dialog associated to the label in order to modify its settings. This item is active only when a right-click is performed on a label.
Remove	Removes the selected labels. To select multiple labels, hold down CTRL during selection. This item is active only when a right-click is performed on a label.
Jump To	Jumps to the step in the Steps panel where the specified label is located. This item is active only when a right-click is performed on a label.

Menu Item	Description
Stay On Top	If enabled, after undocking this panel, it will stay on top of all other undocked panels. This option is disabled by default.
Dockable	If enabled, allows the panel to be docked to its base once it is undocked. If disabled, you will not be able to re-dock the panel once it is undocked. This option is enabled by default. You can undock a panel by holding down and dragging its title bar away from the main window or double-clicking the title bar.
Help	Opens the help topic regarding this debug tool.

Debug Panel - Attachments

Attachments are files of any type that are wrapped and accessible from an Automate Desktop task. These files can be easily exported or deployed along with their associated task allowing them to be more transportable and manageable. At runtime, attachments are automatically "unwrapped" to a local directory where they can be accessed by the task. The Attachments debug panel displays any files that are currently attached to the task. It also allows you to add a new attachment to the task as well as modify or remove existing attachments. More on Attachments



Parameters

The Attachments debug screen contains the following column headers:

Column header	Description
Name	The name of the attachment.
Attachment	The file name (including extension) of the attachment.

Column header	Description
Attached	The date and time when the file was initially attached to the task.
Description	A user defined description of the attachment (this property is optional, therefore, may appear blank).

Context Menu

Right-clicking anywhere inside the Attachments screen displays a context menu with the following items:

Menu item	Description
Add	Adds a new attachment to the task.
Properties	Opens the **Attachment** dialog to display the properties associated to the selected attachment.
Refresh	Refreshes the Attachments list
Remove	Removes the selected attachments. To select more than one attachment, hold down CTRL during selection. > **NOTE:** When you remove an attachment, be sure to remove all references to the attachment from the task, otherwise the task may fail to execute properly.
Remove All	Removes all attachments that exist in a task. > **NOTE:** When you remove an attachment, be sure to remove all references to the attachment from the task, otherwise the task may fail to execute properly.
Help	Opens the help topic regarding this debug tool.

To attach a file to a task

1. Right-click anywhere inside the Attachments debug panel and select **Add** from the context menu that appears. This opens a dialog titled **Attachment** (shown below).

2. Enter the following information:
    - **Name** - The name of the attachment (this value is what you will use as a placeholder for the file).
    - **Attachment file** - The full path and file name of the attachment file. Use the folder icon to browse to the file you wish to attach.
    - **Description** - An optional description of the attachment.
3. Select **OK**. The new attachment and its properties are displayed in the Attachments Debug panel.

> **NOTE:** Multiple files of any type and size can be attached to a task, however, be aware that the size of the task file expands with the size and number of attachments, which may affect performance. Multiple or large sized attachments used in a single task may slow down task execution.

To remove one or more existing attachments

1. From the Attachments debug panel, select existing attachments you wish to remove.
    - To select multiple attachments, hold down CTRL during selection.
2. Right-click the attachments and select **Remove** from the context menu.
    - To remove all existing attachments, select **Remove All**.

> **NOTE:** After removing an attachment, make sure to remove all references to it in any step parameter to ensure that no attachment related error occurs during runtime.

To modify an existing attachment

1. From the Attachments debug panel, right-click the attachment you wish to modify and select **Properties** from the context menu. This opens the **Attachment** dialog.
2. Make the desired adjustments.
3. Upon completion, click **OK** to save changes and close the dialog.

> **NOTE:** If modifications to the original attachment name is made, make sure to update any step parameters referencing the original name to reflect the new name to ensure that no attachment related error occurs during runtime.

Adding Attachments

Attachments allow files of any type to be embedded within a task so that third-party elements can be incorporated for greater portability. At runtime, attachments are automatically "unwrapped" to the local temp directory where they can be accessed by the task. For example, you can create a task that plays a unique sound when a particular event occurs by attaching the associated sound (`.wav` or `.mp3`) file to it. When deploying or exporting this task to other Automate Desktop clients, the addition of the sound file as an attachment ensures that it will always be present and accessible on any system.

Attachments are added and managed by way of the Task Builder's Attachments Debug Panel. It displays any files that are currently attached to the task along with their original file name and extension.

To attach a file to a task

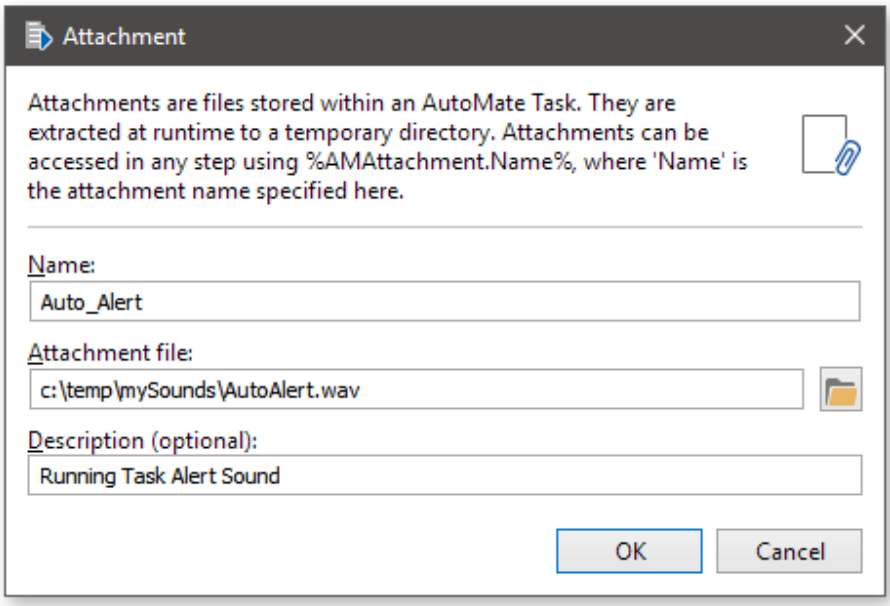
1. Do one of the following to open the Attachment Explorer window:
   - Right-click anywhere inside the **Attachments** debug window and select **Add** on the context menu.
   - Click **Add Attachment** on the **Insert** command group of the ribbon.
   - In the Attachment Explorer window, navigate to, and select the file you want to attach, then click **Open**. This attaches the file to the task and the attachment properties (described below) are displayed in the **Attachments** debug window.
2. In the **Attachments** debug window, view or enter the following information:
   - **Name** - The name of the attachment (required). The name you enter will be used as a placeholder for the file during runtime, therefore, the following rules apply when specifying attachment names:

- Names must contain only alphanumeric characters.
- Must begin with a letter.
- Cannot contain any spaces.

- **Attachment** - The file name of the attachment file. Use the folder icon to browse to the file you wish to attach.
- **Description** - A user defined description of the attachment (optional).
- **Attached** - The date and time the file was initially attached to the task (this value is read-only).

3. Upon completion, click **OK**. The new attachment and its properties are displayed in the **Attachments Debug** panel.

NOTE: Multiple files of any type and size can be attached to a task, however, be aware that the size of the task file expands with the size and number of attachments, which may affect performance. Multiple or large-sized attachments used in a single task may slow down task execution.



Accessing Attachments

Once an attachment is added to a task, it can be accessed in any step by entering:

`%AMAttachment.Name%`

Where `[Name]` is the name of the attachment specified in the **Name** parameter of the **Attachment** dialog.

`AMAttachment.Name` is essentially an Automate Desktop dataset used as a placeholder for the attachment file during runtime. The file itself is unwrapped and saved to the local temp directory. This directory may vary depending on the operating system. On Windows 10, this directory is `C:\Users\[username]\AppData\Local\Temp\attachmentFile.txt`.

Debug Panel - Comments

The Comments debug panel reveals a list of comments included in the task along with the step number in which they are located. You can use this panel to quickly jump to any step that includes a comment. You can also use it to add, modify, or remove one or more comments. More on Comments

Parameters

The Comments window includes the following fields (columns):

Column header	Description
Step	The step number where the comment resides.
Comment	The text description of the comment.

Context Menu

Right-clicking a specific comment or an empty region inside the Comments screen opens a context menu with the following items:

Menu item	Description
Add	Adds a comment to the task. To insert a comment at a specific step, first click the desired step in the Steps panel. The comment is added directly above the selected step. If no step is selected, the comment is inserted at the end (last step) of the task.
Properties	Opens the selected comment's dialog allowing you to view or modify the comment. This item is active only if a comment is selected.
Remove	Removes the selected comments. To select multiple comments, hold down CTRL during selection. This item is active only if a comment is selected.
Jump To	Jumps to the step where the selected comment is located. This item is active only if a comment is selected.
Help	Opens the help topic regarding this debug tool.

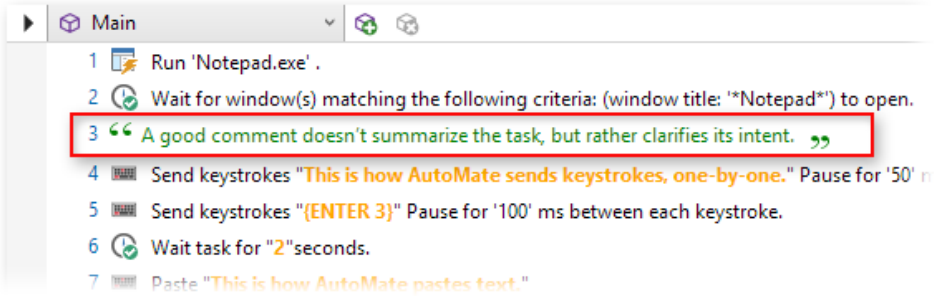
Adding Comments

Comments are used to embed user readable annotations in the steps of a task, usually with the purpose of making the task easier to understand. Comments can be used to summarize the task, explain the intent of a specific step, document instructions, enter notes, reminders or other important information. Comments are ignored by Automate Desktop during runtime. For debugging convenience, the Comments Debug Panel displays each comment embedded in a task and their associated step number.

To add a comment

1. From the Ribbon's **Insert** command group, select **Add Comment** or press CTRL+ALT+C. A new Comment step is added in the Task Builder's Steps panel at the end of your task.
2. Click inside the **Enter a comment...** text-box and enter your comment. The new comment is colored green by default, however, you can change the default color by way of Color Options.

3. Use the **Up** or **Down** arrows to move the comment to the desired step or use your mouse to drag and drop the comment to the desired step.



> **NOTE:** Comments can also be added from the Debug panel by clicking the **Comments** tab, right-clicking inside the panel, and then selecting **Add** from the context menu.

Debug Panel - Call Stack

A stack is a dynamic data structure that stores information about the active sub-tasks contained in a task. The Call Stack debug window changes dynamically when sub-tasks are used. It displays execution depth of the current task and allows you to keep track of the point at which each active sub-task should return control to the main task when it finishes executing.

Parameters

The Call Stack debug window generates information into a single column:

Column header	Description
Name	The name of the current task and task function are enclosed inside [] brackets followed by the current step number (for example, `[TheTaskName::Main] Step #10`). **NOTE:** If no task function is selected, the default task function named "Main" will be displayed.

Context Menu

Right-clicking inside the Output panel opens a context menu that contains the following items.

Menu item	Description
Jump to	Jumps to the current step in the Steps panel. **NOTE:** Use a breakpoint to pause the task at a specific step.
Help	Opens the help topic regarding this debug tool.

Debug Panel - Regions

A region is a named section of task steps that can be collapsed as required in order to hide their contents. The use of regions can effectively group a task into sections, thus, can be useful for organizing lengthy tasks and making complex tasks more comprehensible. You can easily expand a region at a click of a button to access steps that you need to work on and collapse the region upon completion. The Regions debug panel lists existing regions within a task, including the name of the region, its starting step and ending step. It also allows you to jump to a particular region, clear a specific region or clear all regions that exist in the task. More on Regions

Parameters

The Regions debug window generates information divided into the following columns:

Column Header	Description
Starting Step	The step where the selected region starts.
Ending Step	The step where the selected region ends.
Region	The name of the selected region.

Context Menu

Right-click anywhere inside the Regions window to view the following properties:

Menu Item	Description
Jump To	Jumps to the step where the selected region is located.
Clear Region	Removes the selected regions. To select multiple regions, hold down CTRL during selection.
Clear All Regions	Removes all existing regions.
Help	Opens the help topic regarding this debug tool.

About Regions

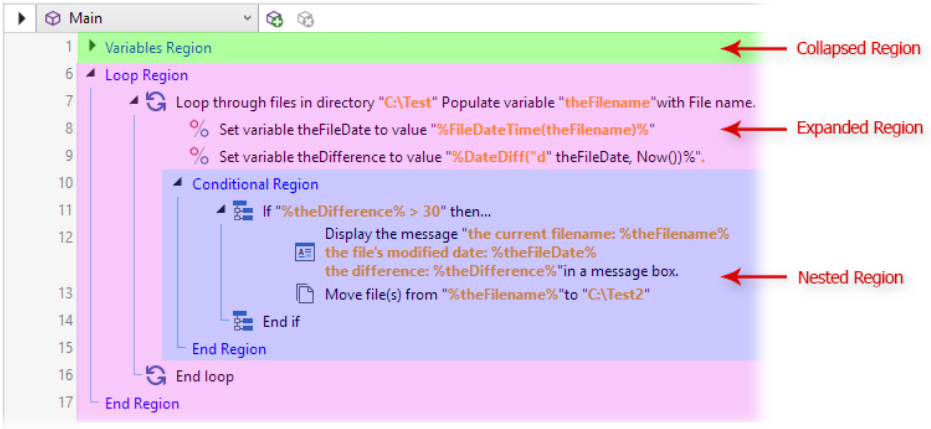
A region is a uniquely named section of a task that can be hidden from view as required. This allows areas of a task to be organized into separate segments, thus, making it less cluttered and more manageable. When a region is collapsed, contents are hidden from view and the region appears as a single step in the task. When a task runs from Task Builder, collapsed regions expand automatically so the developer can view each step as it executes. Upon task completion, regions are re-collapsed to their former state. Automate Desktop supports nested regions (region within a region) to further expand their capabilities.

Regions have no effect on the final, developed task nor does it influence its execution. They only affect the manner in which task steps are visually displayed in Task Builder. Regions can be created to categorize blocks of steps in a task based on common functionality or actions. A unique name can be entered for a region as a way to ascertain its contents when it is collapsed. If a particular step inside a region needs to be viewed or modified, simply expand the region that contains that step while the rest of the regions remain hidden from view.

The Regions debug panel lists existing regions within a task, including the name of the region, its starting step and ending step. It also allows you to jump to a particular region, clear a specific region or clear all regions that exist in the task.

Creating and Populating Regions

You create a region by selecting one or more existing steps and clicking the **Create Region** command on the ribbon or context menu. When a region is created, its default name is always 'New Region.' It is important that you enter a unique name for newly created regions in order to avoid confusion. A collapsed region is designated by a down arrow symbol in the margin near the region. The down arrow symbol can be clicked to expand the region. Once expanded, the down arrow is replaced with an up arrow and a line terminating at the end of the region. The up arrow can be clicked to re-collapse the region. The image below displays an example of a collapsed and expanded region.



To create a region

1. Highlight the steps in the Steps panel you wish to place in a region. To highlight multiple steps, hold down CTRL during selection (selected steps must be sequential).
2. Do one of the following:
    a. Click the **Create Region** button located on the ribbon's **Step** command group.
    b. Right-click and select **Region** > **Create Region** from the context menu that appears.
    c. Press CTRL + R on your keyboard.

A new region is added to the Steps panel populated with the steps you initially selected with the default name supplied for the region highlighted. Simply type over the default name of the region to give it a unique name.

To add a new action or activity to an existing region

1. Expand the region you wish to add an action to by double-clicking the region name or clicking the expand symbol to the left of the region name.
2. Drag the desired action or activity from the Actions panel onto the desired area of the region. A blue line appears in the Steps panel signifying the section in which the action or activity will be placed.
3. Release the mouse button when the blue line arrives at the section of the region you wish to drop the action or activity.

To move existing steps onto a region

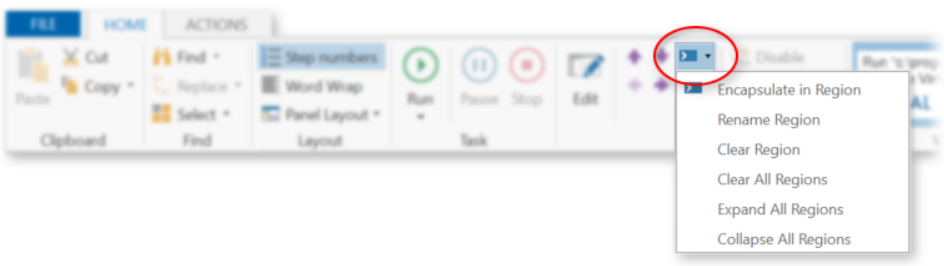
1. Expand the region you wish to move existing steps to by double-clicking the region name or clicking the expand symbol to the left of the region name.
2. Highlight the steps that are currently outside of the region you wish to include. To highlight multiple steps, hold down CTRL during selection (selected steps must be sequential in order).
3. Drag the steps to the desired part of the region. A blue line will appear signifying the section of the region in which the steps will be placed.
4. Release the mouse button when the blue line arrives at the section of the region you wish to drop the steps.

To create a nested region (region within a region)

1. Expand the region you wish to add a sub-region to.
2. Highlight the steps within that region that you would like to place in the sub-region.
3. Click the **Create Region** button from the Task Builder Ribbon's **Step** command group or right-click and select **Create Region**. A new sub-region is added inside the parent region.
4. Click the newly created sub-region to rename it.

Managing Regions

Existing regions can be expanded, collapsed, renamed, modified or cleared. For example, when you are finished constructing your task, you may opt to remove all existing regions with the **Clear All Regions** command. All region related commands are located on the ribbon's **Home** tab in the **Step** command group (circled below) or by way of a context menu that appears when you right-click a region inside the Steps panel.



To expand all regions simultaneously

- Select **Expand All Regions** from the Ribbon's **Step** command group or right-click menu.

To collapse all regions simultaneously

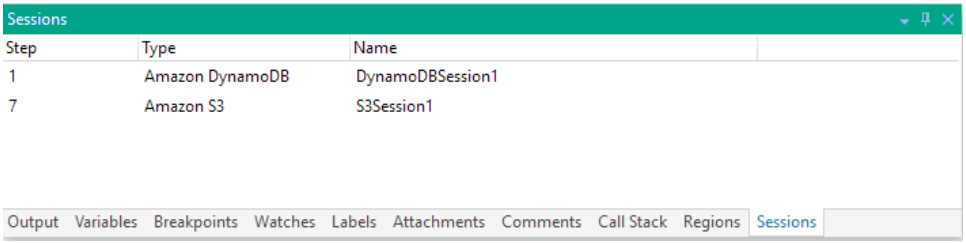
- Select **Collapse All Regions** from the Ribbon's **Step** command group or right-click menu.

To clear one or more regions

1. Right-click the step that contains the region's name and select **Clear Region**.
2. To clear all existing regions, right-click any step that contains a region's title and select **Clear All Regions**.

Debug Panel - Sessions

The Sessions debug panel can be used to display a list of steps that contain sessions within the task.



Parameters

The Sessions debug window includes the following fields (columns):

Column header	Description
Step	The step number where the session resides.
Type	The action that the session derives from.
Name	The name of the session.

Menu Items

Right-click inside the Sessions window to view the following menu items:

Menu item	Description
Jump To	Allows you to jump to the specified session.
Help	Opens the help topic regarding this debug tool.

Advanced Workflow Module Features and Benefits

Advanced Workflows is a powerful tool that allows rapid construction of Windows automation routines via drag and drop without any programming. It replaces legacy batch files, scripts and custom application development, enabling complete IT Process automation without writing a single line of code. Advanced Workflows provides an incredibly easy-to-use, intuitive interface for developing automation applications. The basic building blocks are called actions, which are plain-English, drag-and-drop, fill-in-the-blank tools developers use to build Tasks. Developers simply drag-and-drop actions in Task Builder to build a series of Steps which collectively make up the task.

Available Actions & Activities

Includes 100s of actions and activities that involve extensive support for FTP/SFTP, SQL, Email, HTTP, VB Scripts, VMWare Host and VMWare Guest, Services, Environment Variables, Exchange, MSMQ and many more. This list does not begin to address all the possible automated solutions that can be constructed and deployed. A task can contain any number of actions/activities in any order, creating limitless possibilities.

Intuitive Task Development & Debugging

The Task Builder is an intuitive interface for developing automation applications. It is used to visually construct and examine the steps that a task should carry out when it is run. To ensure that newly created tasks are working properly before they are put into production, the Task Builder also includes a variety of testing and debugging features that outputs real-time information about a running task. They enable monitoring of each step during task execution, allow inspection of variables, datasets and other dynamic data and aid developers with troubleshooting important aspects of a task as it is being constructed.

Variable Support

Includes full support for variables that are used to pass dynamic data during task execution and also supports one and two-dimensional arrays for implementing more complex logic to tasks. Additionally, many actions support the creation of datasets, which can hold multiple rows and columns of dynamic data. This is useful when retrieving data that describes a collection of information such as a database or spreadsheet.

Replaces Code

Reduces costs and hassles associated with developing and maintaining code, scripts, and batch files. For developers, increases the speed of development without sacrificing power, saving time and future code maintenance hassles. For non-developers, provides the simplicity to complex processes without the need for code or syntax.

Ease of Use

Provides an incredibly user-friendly, yet intuitive set of interfaces allowing you to visually create, manage, and execute your automated tasks without requiring any programming expertise. Instead of writing scripts and batch files, business and IT processes are created using the Task Builder interface which contains over 300 prebuilt actions and activities that you can easily drag-and-drop together to form powerful automated tasks.

Saves Time & Reduces Errors

Network managers and IT administrators can eliminate time spent on repetitive IT processes so they can focus on other aspects of their jobs. Additionally, automation

eliminates errors introduced during manual processes that often result in network downtime, delayed access to data, and user complaints. Streamline inefficient processes and operations with reliable, repeatable, automation and eliminating errors introduced by process delays and repetitive worker activities.

Accelerates flow of information

Drag-and-drop building enables tasks to be constructed rapidly. This accelerates data flows, allowing business users and decision-makers to become more effective and efficient by delivering real-time access to accurate data.

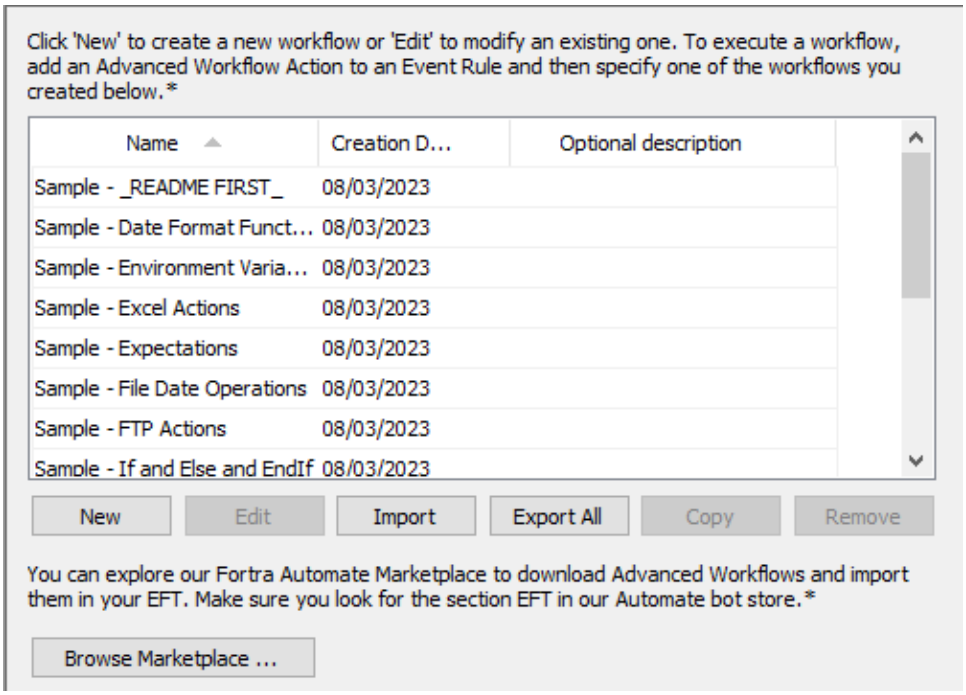
Creating Workflows for Use in an Event Rule

(Requires Advanced Workflow Module) Similar to Commands, Workflows are used in Event Rules as Actions or triggers. When you create a Workflow, it is saved in the SQLite database file in **C:\ProgramData\Globalscape\EFT Server**.

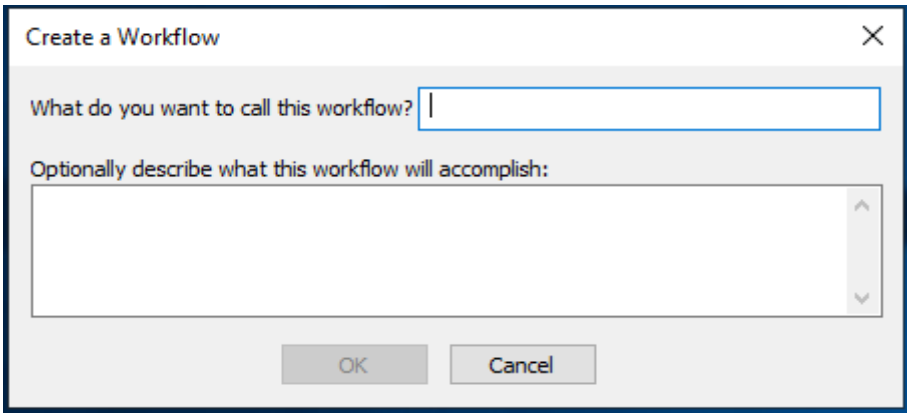
During the Advanced Workflow trial, when a new Workflow is created, a message appears (prior to the **Create a Workflow** dialog box) informing you that the Advanced Workflow module is an optional module and that the trial begins when the first Workflow is created.

To create a Workflow

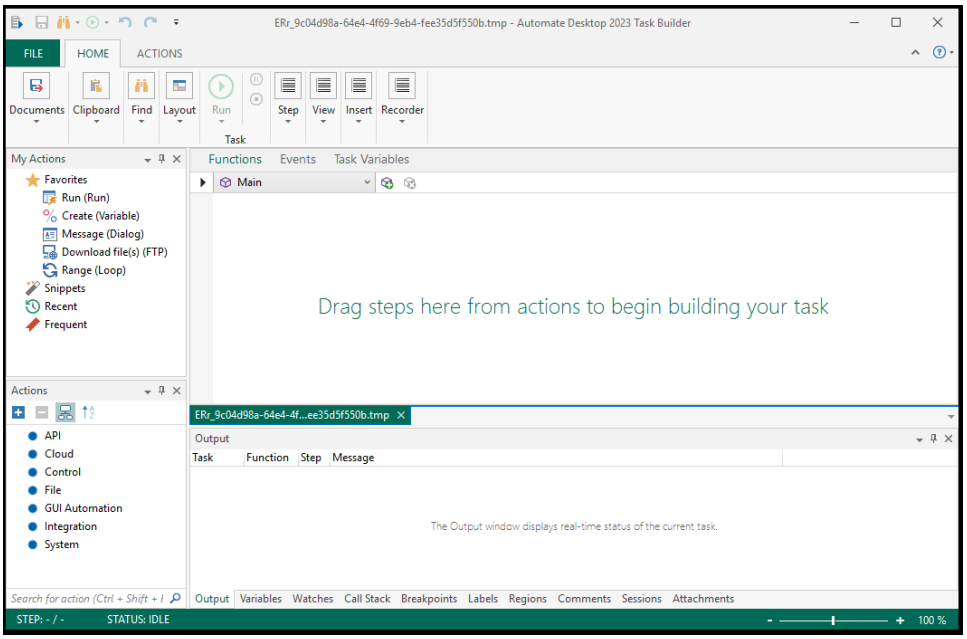
1. In the administration interface, connect to EFT and click the **Server** tab.
2. On the **Server** tab, click the **Advanced Workflows** node.
3. In the right pane, the **Advanced Workflows** tab appears.



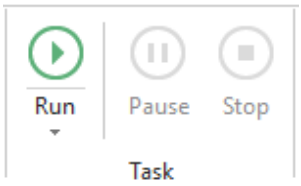
4. In the right pane, click **New**. The **Create a Workflow** dialog box appears.



5. In the **What do you want to call this workflow** box, specify a name for the Workflow. When you add the workflow to Event Rules, the name you specify here appears in the Rule.
6. (Optional) Provide a description of the Workflow, and then click **OK**. The Workflow **Task Builder** appears.



7. The tree in the left pane lists the steps that you can add to the Workflow. The right pane displays the steps in the Workflow.
8. Drag items from the **Actions** list to the **Steps** pane to create your Workflow.
9. Use the **Run** icon on the Debug toolbar to test the steps. You can run the whole Workflow all at once, run only a selected step, or run the whole Workflow starting with a step other than the first step.

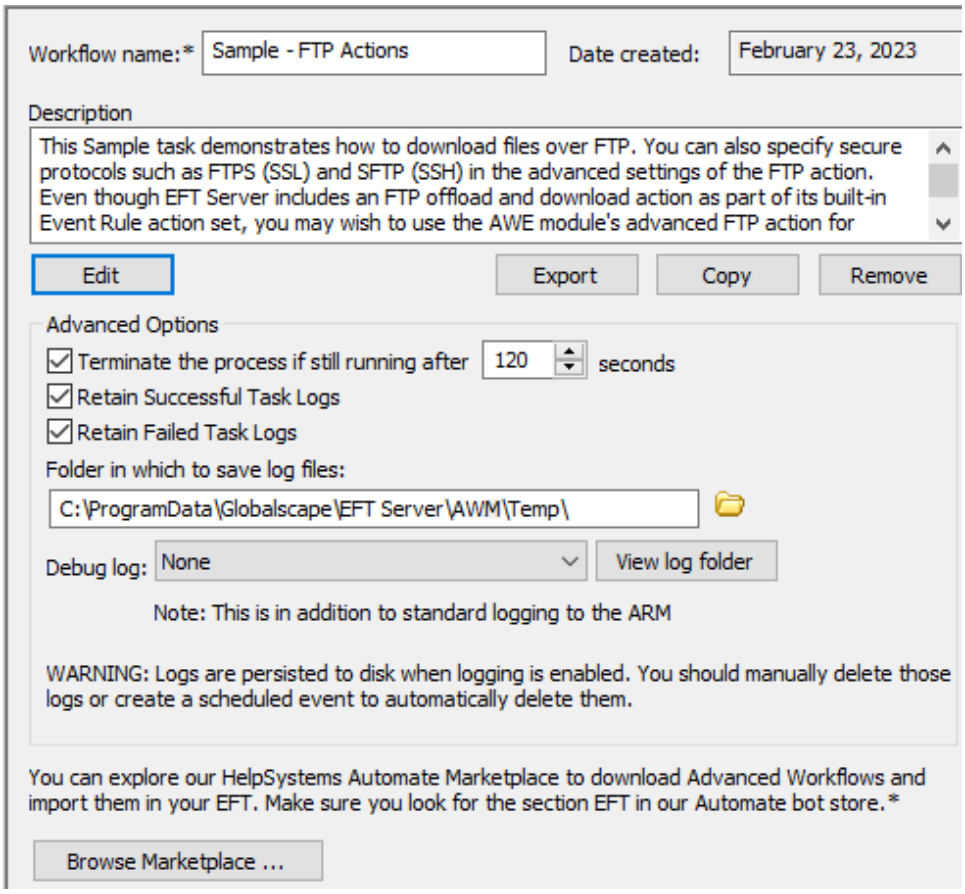


The Output pane displays the result of each step. For example:

```
Executing line 5
Starting Input Box with message "What is your name?"...
Creating message box "What is your name?"...  >
Populating variable "theUserName"...
```

```
Finished Input Box "What is your name?".
The step was okay.
```

10. After you have created your Workflow, click **Save and Close**. The Workflow appears in the **Advanced Workflows** pane of the Site and is ready to be used in Event Rules.



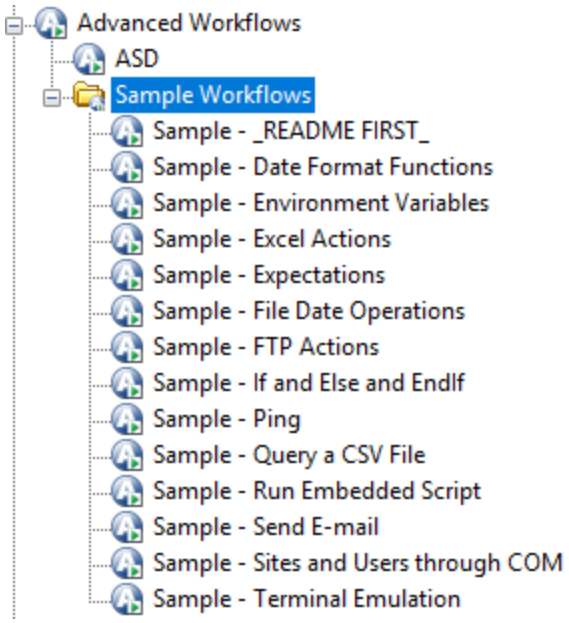
11. (Optional) In the **Advanced Options** area:
    - Select the **Terminate the process** check box and specify the number of seconds after which to terminate the Workflow if it fails to execute.
    - Select the **Retain Successful Task Logs** check box if you want to keep all successful attempts for this Workflow
    - Select the **Retain Failed Task Logs** check box if you want to keep all failed attempts for this workflow

- Specify the location in which to save logs, if different than the default of C:\ProgramData\Globalscape\EFT Server\AWE\Temp\ (for example, in a shared location)
- Specify the level of debug logging in the **Debug log level** box, **None**, **Minimal**, **Normal**, or **Verbose** (None is the default).
- Click **View log folder** to view the CSV logs created by this workflow, saved in <installation_folder>\AWE\Temp. If you enable the logging, you should manually delete the files after you're done with them or create a Scheduled event in EFT to delete them automatically.

Your Workflow is now ready to insert into an Event Rule. The Auditing and Reporting module Event Rule reports will show the Advanced Workflow task name.

Creating Advanced Workflow Folders

Create folders in the Advanced Workflow node to organize workflows. For example, you might want to create a "Sample Workflows" folder, then move the Sample workflows into the new folder.



To create an Advanced Workflow folder

1. In the administration interface, connect to EFT and click the **Server** tab.
2. On the **Server** tab, right-click the **Advanced Workflows** node, then click **New Workflow Folder**.
3. Provide a name for the folder (max value of 260 characters), keeping the name and characters simple to adhere to Windows path requirements. For example, name it "Sample Workflows" and copy the Sample workflows there.
4. One at a time, click the workflow to move, then click-and-drag it to the new folder. (Multiselect is not yet available for workflows.)

> **NOTE:** This functionality is also available in REST API. (Delete Sites AWE Task Folders, Get Sites AWE Task Folders, Patch Sites AWE Task Folders Post Sites AWE Task Folders).

Adding a Workflow Action to an Event Rule

(Requires the Advanced Workflow module.) With Advanced Workflow Actions, EFT does not wait for a reply before returning control to the Event Rule thread, *unless* an "if failed" Action was specified, such as **Stop Processing this Rule**, in which case the Action waits for a return message indicating success or failure from the invoked process.

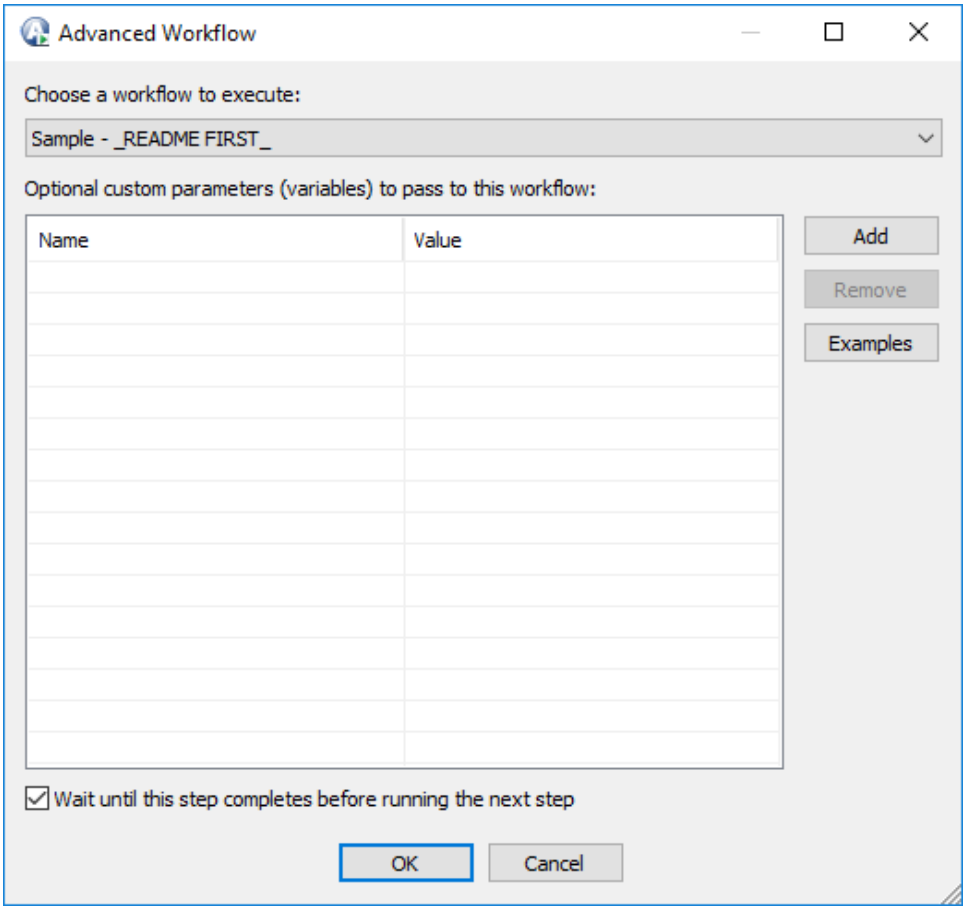
The workflows created for use in Event Rules are executed using the EFT server administrator credentials, unless you specify otherwise. If a resource cannot be accessed using the credentials under which the EFT service is running, you need to include the **optional credentials for the destination server.**

- Think of "Local Transfer" as an operation (offload or download) with a remote server.
- Think of "Optional credentials override" as "credentials to access remote server."
    - For download action, it is "credentials for source folder."
    - For copy/move (offload), it is "credentials for destination folder."

- "Credentials to access local folder" ("source" for offload and "dest" for download) is Event Rule execution context (EFT account, or Folder Monitor account for FM rules, or Connected Client account for client-originated rules on an AD site):
    - Offload: local (EFT) => remote ("override credentials")
    - Download: local (EFT) <= remote ("override credentials")
    - TEST1: Offloads file from "local" Share A (access as EFT account, i.e., X) to "remote" folder B (access as Y) => Fails, as X has no permissions on A.
    - TEST2: Downloads file from "remote" Share A (access as Y) to "local" folder B (access as EFT account, i.e., X) => Succeeds, as Y has permissions on A and X has permissions on B.

To add a Workflow to an Event Rule

1. Create a Workflow.
2. Create an Event Rule.
3. In the **Actions** list, double-click **Script: Advanced Workflow**. The Action is added to the Rule.
4. In the **Rule Builder**, click the **Advanced Workflow** link. The **Advanced Workflow** dialog box appears.



5. The defined Workflows appear in alphabetical order in the **Choose a workflow to execute** list (at the top of the **Advanced Workflow** dialog box). Click the down arrow to select a Workflow.
6. (Optional) Specify custom parameters to pass to the Workflow in the **Name** and **Value** columns, and then click **Add**.

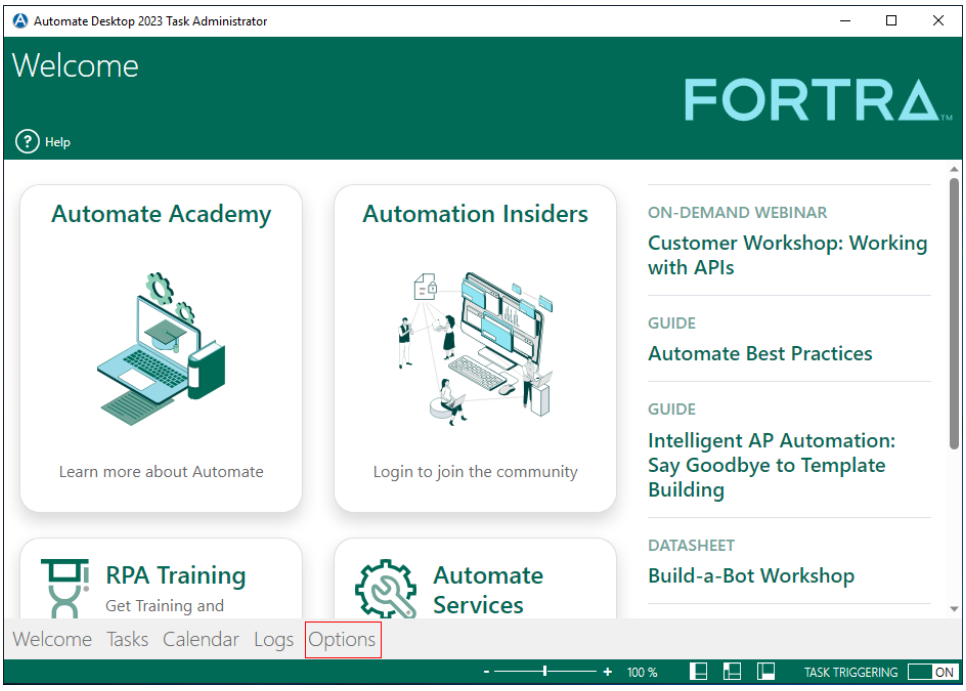
7. (Optional) The **Wait until this step completes before running the next step** check box is selected by default.
    a. When the check box is selected, you can select the "If failed" action, and populate it with actions to run in case the rule fails.
    b. To allow the next step to run before this action completes, clear the check box. If check box is not selected, a prompt appears to confirm: "This step will be executed asynchronously (non-blocking), which means EFT won't wait for this step to be completed before running the next step. This could yield undesirable results if the next step depends on the output or outcome of this one. Are you sure you want to make this action asynchronous?" (All actions in the IF FAILED section are lost if the parent action is switched from async to sync mode.)
8. Click **OK**. The **Advanced Workflow** link in the **Rule Builder** updates with the name of the Workflow.
9. Add other Actions as needed, and then click **Apply** to save the changes on EFT.

Enable or Disable Running Task Window

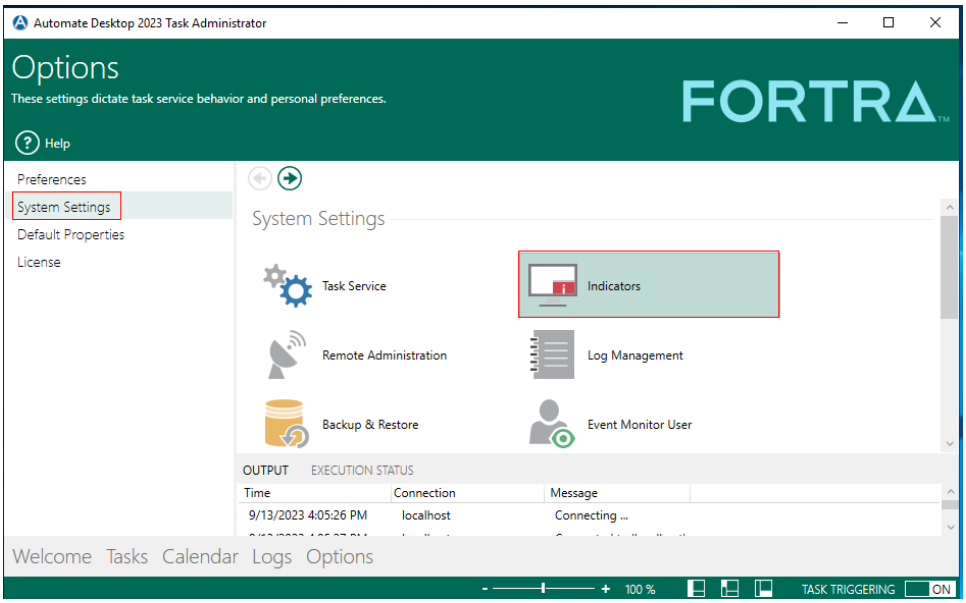
Certain Automate Actions trigger a "Task window" pop-up message that appears in the bottom right corner of the display. EFT event rules run unattended, therefore, pop-up dialog boxes are unnecessary. This functionality can have some issues with EFT environments that have multiple event rules that are triggering at any given time with continuous popups being displayed. You can enable or disable the "running task" popups in the Automate Task Administrator, as described below.

To enable or disable the running task window

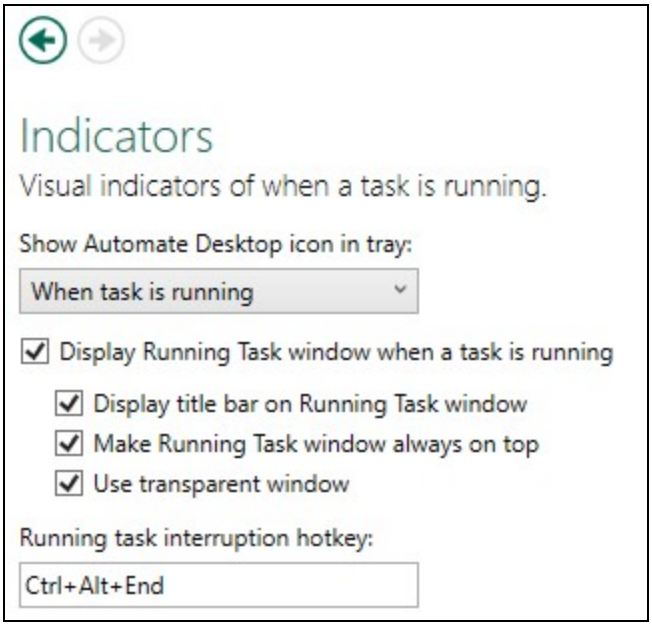
1. Open the Task Administrator: **Start > Automate Desktop 2024 > Automate Desktop Task Administrator**.



2. In the bottom menu, click **Options**.



3. In the upper left, click **System Settings**.
4. In the center of the display, double-click **Indicators**. The **Indicators** options appear.



5. Under **Show Automate Desktop icon in tray:**
    - Select (enable) or clear (disable) the **Display Running Task window when a task is running** check box.

-OR-

- Click the drop-down list and click **Never** instead of **When a task is running**.

6. Click **Apply**.

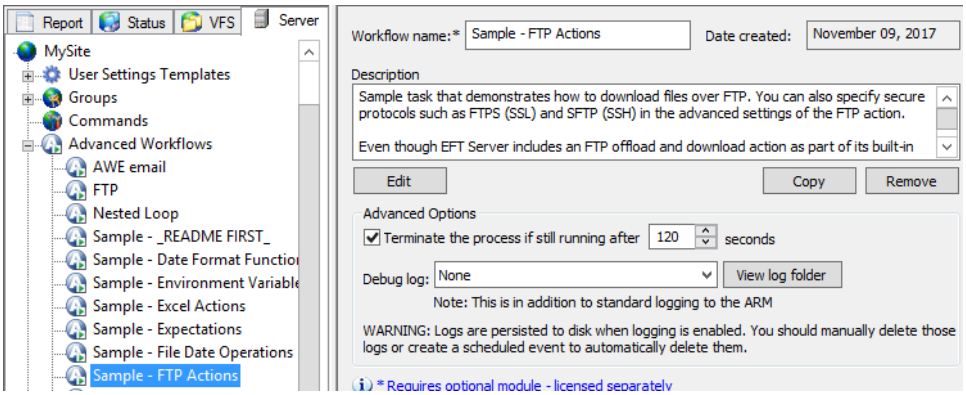
Sample Workflows

The Advanced Workflows module includes several samples to demonstrate how to create a workflow. The comments in the workflow provide instructions. The sample files are stored in **C:\ProgramData\Globalscape\EFT Server** in a database file named SiteConfig.<GUID> file (one for each Site).

> **NOTE:** In the Advanced Workflows module, variables cannot contain periods; therefore, in each variable that contains a period, the period is replaced with an underscore. For example, change `%CONNECTION.LOCAL_IP%` to `%CONNECTION_LOCAL_IP%`

To view the sample workflows

1. In the EFT administration interface, connect to EFT and click the Server tab.
2. In the left pane, expand the Site node for the Site that you want to configure, then click the Advanced Workflows node. The node expands to show the Sample Workflows.



3. In the left pane, click a sample workflow. The right pane displays the properties of the selected workflow.

4. Do one of the following to open the workflow in the Task Builder:
    - In the right pane, click **Edit**.
    - In the left pane, double-click the workflow.
5. View the comments in the Steps pane for instructions on how to configure the workflow. Use this guidance to create similar workflows.
6. If you want to save the sample workflow with your changes, click **Save and Close**. The workflow is saved in **C:\ProgramData\Globalscape\EFT Server** in a database file named SiteConfig.<GUID> file.

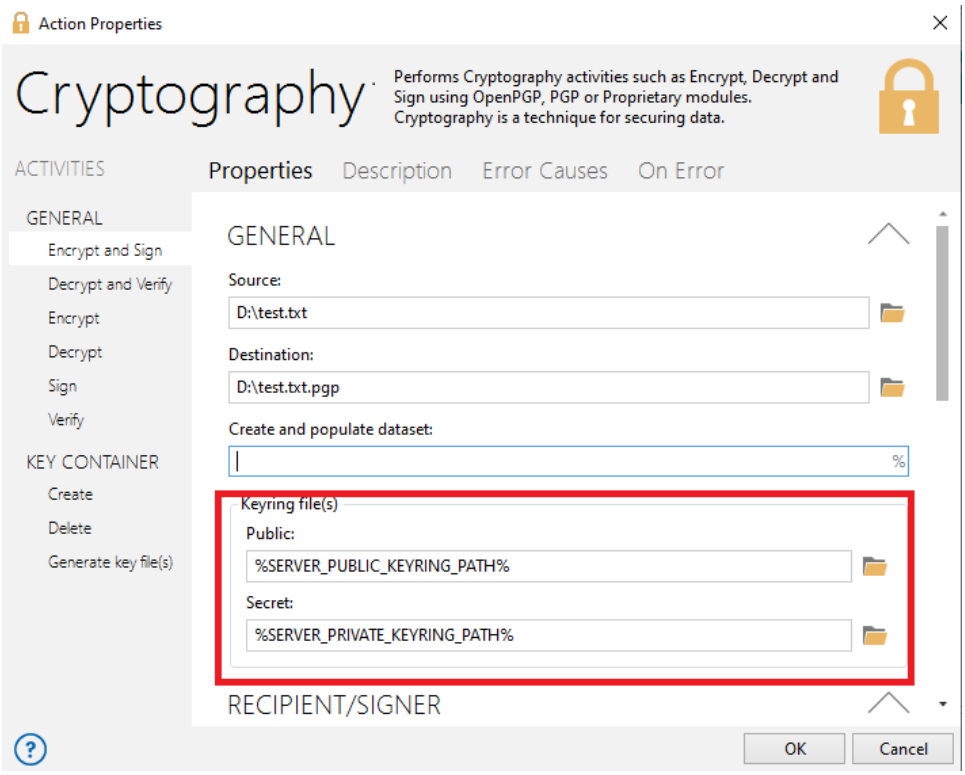
If you have accidentally overwritten a sample workflow and you want to revert to the original version of the sample file, you can copy the original from the default location (**C:\ProgramData\Globalscape\EFT Server\AWE**). You could also make a backup copy so you don't lose the original.

Advanced Workflow Variables

Used when the Execute Advanced Workflow Action is added to an Event Rule.

Text Displayed	Variable	Description
Advanced Workflow Name	%AWE.TASK_NAME%	Workflow name
Advanced Workflow Log Path	%AWE.LOG_PATH%	Workflow log path
Advanced Workflow Error Code	%AWE.ERROR_CODE%	Workflow error code
Advanced Workflow Error Description	%AWE.ERROR_DESCRIPTION%	Workflow error description
Advanced Workflow Error Line	%AWE.ERROR_LINE%	Workflow error line
Advanced Workflow Result Code	%AWE.RESULT_CODE%	Workflow result code displayed in the Windows Event Log and others
Advanced Workflow Result Description	%AWE.RESULT_DESCRIPTION%	Workflow result description displayed in the Windows Event Log and others
Advanced Workflow Execution Time (ms)	%AWE.EXECUTION_TIME_MS%	Workflow execution time (ms)

You can also pass other EFT variables into the Advanced Workflow Actions. For example, the path to the OpenPGP keyring files can be passed to the workflow with the %SERVER_PUBLIC_KEYRING_PATH% and %SERVER_PRIVATE_KEYRING_PATH% variables.



Advanced Workflow Conditions

You can apply Advanced Workflow Conditions to any events except the Cloud-Based events.

To use one or more Advanced Workflow Conditions

1. Create the Event Rule.
2. Double-click the Condition, or click the Condition, then click **Add Condition**.
3. In the **Rule Builder**, click the hyperlinks in the Condition to specify if the Condition **does** or **does not** and **less than**, **equal to**, or **greater than** the [code], [test], or [number].
4. Then add one or more Actions and click **Apply**.

> **NOTE:** The Advanced Workflow Conditions below are available for any events except cloud-based events. They are named for their function. For example, the "If Advanced Workflow Error Description" condition will trigger the event if the Error Description in the Advanced Workflow has or doesn't have the text specified.

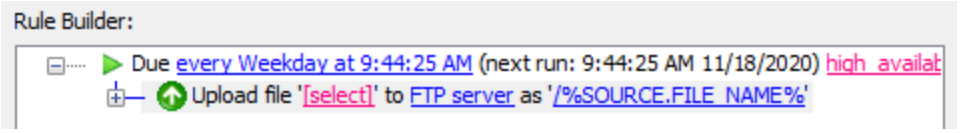
- If Advanced Workflow Error Code does, does not, less than, equal to, or greater than [code]
- If Advanced Workflow Error Description does, does not, equal to [text]
- If Advanced Workflow Error Line does, does not, less than, equal to, or greater than [number]
- If Advanced Workflow Result Code does, does not, less than, equal to, or greater than [code]
- If Advanced Workflow Result Description does, does not, equal to [text]
- If Advanced Workflow Execution Time does, does not, less than, equal to, or greater than [ms]

Backing Up Advanced Workflows

If you plan to edit the sample Workflows and/or create custom Workflows, you can create an Event Rule to periodically back up (save a copy of) the Workflows with a Timer rule and an Upload action.

To backup the Workflows

1. Define a Timer Rule. Specify the frequency depending on how often you create new Workflows.
2. Add the Upload action to the Rule. (Refer to Protocol: Upload Action for details.)



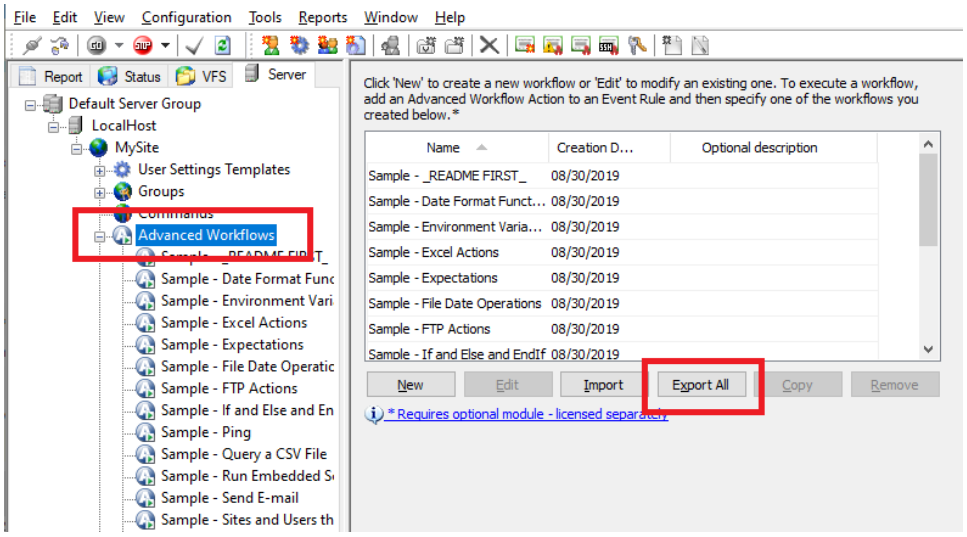
3. In the **Rule Builder**, click one of the undefined parameters (for example, '[select]' or '/%SOURCE.FILE_NAME%'). The **Upload/File Offload Configuration** wizard appears.
4. In the **Source** path box, specify the location of the Workflow (**.aml**) files. For example, to copy all of the Workflows for the Site named "MyGSSite," in the **Source path** box type
   `C:\ProgramData\Globalscape\EFT Server\AWE\MyGSSite_?.*`.
   If you use *, you will backup everything in that folder.
   (Do ***NOT*** select the **Delete source file** check box!)
5. In the **Destination** path box, specify a location on a remote drive (in case the local drive fails).
6. Click **Finish** in the wizard.
7. Click **Apply**.

Importing and Exporting Advanced Workflows

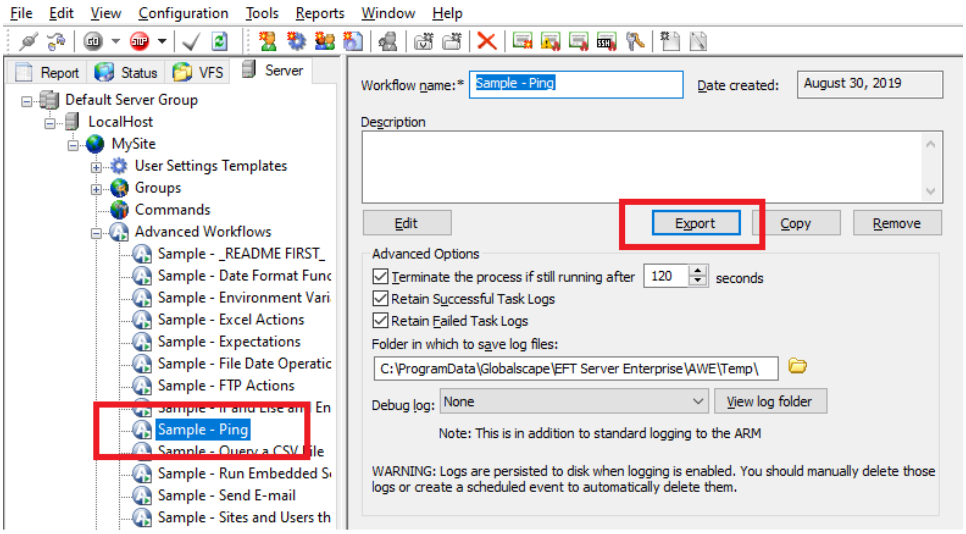
When moving an installation of EFT from staging to production, the biggest issue is moving the Event Rules. You can export the Workflows for source control, "diff" reports, interaction with support, manual editing, migrations, and so on.

> **NOTE:** After upgrading to EFT v8.2, tasks (AML files) imported from older versions of Automate (v10 or older) will be converted to Automate 2024 when imported using the **Import** button on the Advanced Workflows node interface. Imports that fail will be logged in an Advanced Workflow Module migration log with information about the conversion process so that the administrator can be aware of any issues or changes detected during conversion. If you try to import an AML with the same name as an existing AML, the **OK** button in the **Import Workflow** dialog box will be dimmed (unavailable).

You can export all of the workflows ...



... or just the one that you have selected:

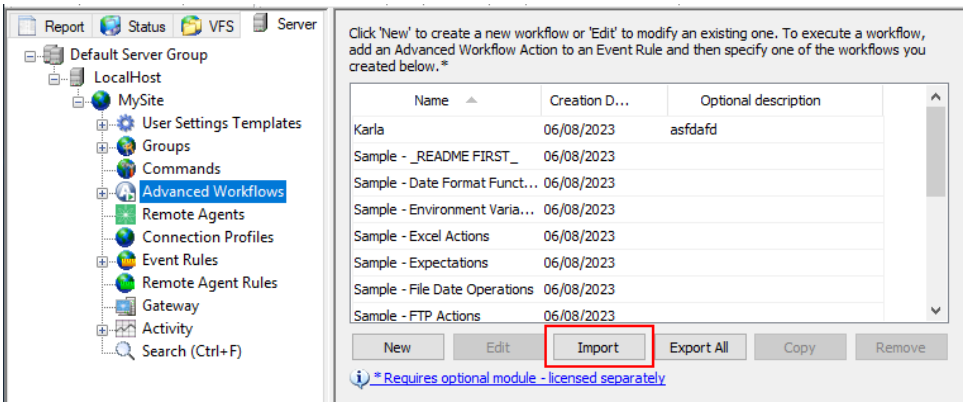


To export Advanced Workflows

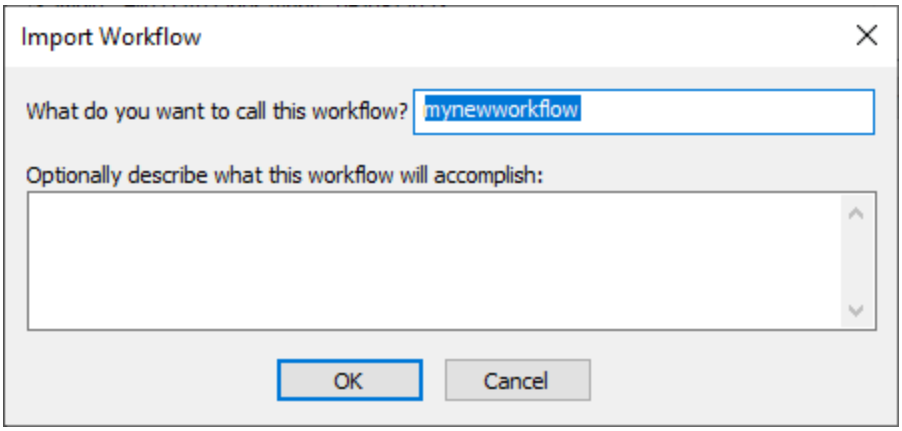
1. Click the **Advanced Workflows** node or the specific workflow that you want to export, then click **Export**. The **Browse for Folder** dialog box appears.
2. Browse for or create the folder in which you want to save the exported files, click **OK**. The AML files are saved in the specified folder.

To import Advanced Workflows

1. Click the **Advanced Workflows** node.
2. In the panel to the right of the tree, under the **Description** box, click **Import**.



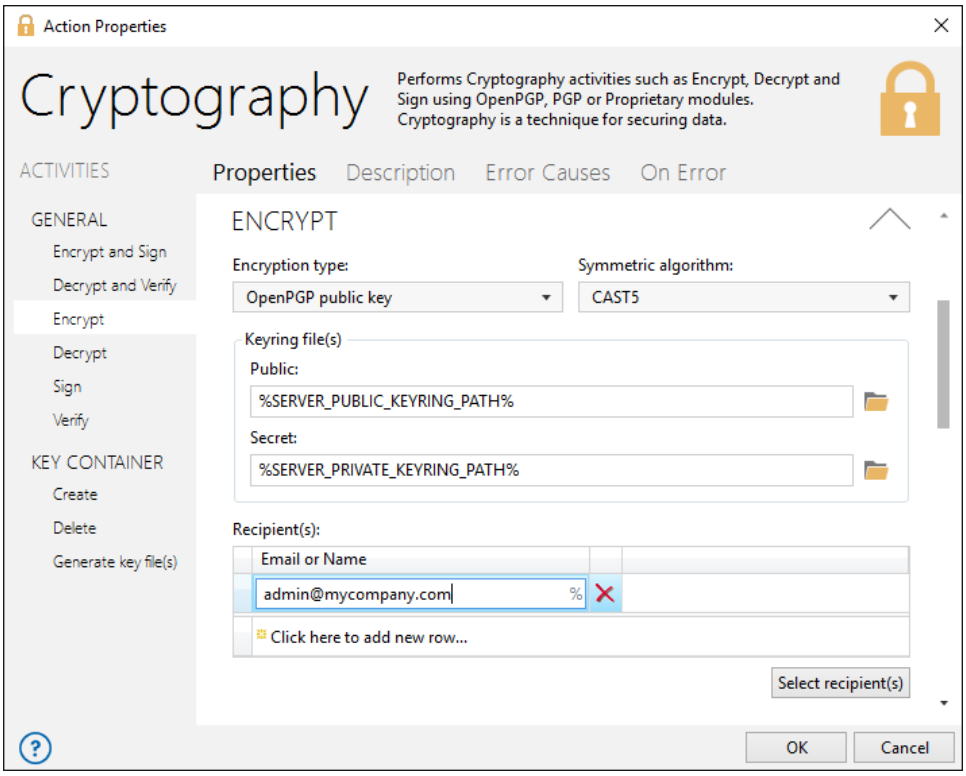
3. In the **Open** dialog box, browse for the AML file that you want to import, then click **Open**.The **Import Workflow** dialog box appears.



4. The name of the file that you imported appears in the **What do you want to call this workflow?** box. You can keep that name or change it.
5. (Optional) Provide a description of the workflow, then click **OK**.
6. The imported workflow appears under the **Advanced Workflows** node.

Encrypt Files in a Workflow Using EFT OpenPGP Key

The EFT OpenPGP keys are stored in the SiteConfig.<GUID> file. If for some reason you want to reference the keys in an Advanced Workflow instead of the EFT OpenPGP Action, you can use the Advanced Workflow Cryptography action. You can then test it in the EFT event rule (not in the Task Builder).



To encrypt files using EFT OpenPGP keys

1. Pass in the public and private keys using EFT variables (rather than an actual path): %SERVER_PUBLIC_KEYRING_PATH% and %SERVER_PRIVATE_KEYRING_PATH%.
2. Manually add the recipient by clicking **Click here to add new row** and then typing the email address that matches the one in the keyring.
    - Do NOT type **Select recipients** to find the address.
    - Do NOT test the Workflow in the Task Builder.
3. Click **OK** to save the action in the Workflow.
4. Add the Workflow to an EFT Event Rule in the EFT Rule Builder.

Technical Reference

This section provides help for the following features:

The Technical Reference in the Task Builder provide a reference for the following components:

- **AML (Automate Markup Language)** - similar to XML, AML is the primary internal language used by Automate to define and execute tasks. When you create a task using the Task Builder, you are creating a block of AML.
- **AMError** - When a task generates an error, it automatically creates a dataset called AMError, which contains details about the error.
- **AMTask** - When a task object executes within a workflow, an Automate dataset uniquely named AMTask is automatically created. This dataset contains a collection of task-related data.
- **AMTrigger** - When a task is triggered, it automatically creates a dataset called AMTrigger. This dataset contains global information about the trigger (task name, trigger date and time, computer, etc.) as well as trigger-specific datasets containing detailed information about that status that caused the trigger.
- **Basic Scripting** - Based on Visual Basic, the Basic Scripting engine extends the functionality provided by AML, giving you access to the Windows API, COM, OLE, and ActiveX objects, and a large number of functions.

Introduction to AML (Automation Markup Language)

AML (Automation Markup Language) is the primary internal language. It is a markup language similar to HTML or XML and defines a set of rules for encoding the structure, layout and contents of tasks. AML is powerful and scalable, yet its format is designed to emphasize simplicity and usability, allowing data to be easily interpreted and shared amongst experienced programmers and novices alike. AML is comprised of a variety of available actions/activities and hundreds of functions and instructions with built-in support for variables, constants and expressions.

> **NOTE:** Do not let the fact that there is an underlying "language" scare you away. The Task Builder provides an easy-to-use visual interface layer over the top of AML. It is not necessary to learn the actual internal syntax of any of these commands; this information is documented for advanced users that wish to familiarize themselves with AML. .

BASIC Scripting encompasses all available functions which may be used either as expressions inside any task step by surrounding the function in % signs, or inside a BASIC script by using the BASIC Script action. It is not necessary to use BASIC scripting to use AML. BASIC scripting is only available to extend the built-in capabilities of AML.

The code below displays a simple task as it would appear in AML format. It is comprised of 3 steps:

1. A comment step which can be used to describe what the task will do.
2. Creates a variable with the value "Hello World!"
3. Displays the variable's value in a message dialog.

```
<!--This is an optional comment. It can be used to describe this
task. -->
<AMVARIABLE NAME="Name_of_Variable" VALUE="Hello World!"
DESCRIPTION="Description of Variable" />
<AMSHOWDIALOG MESSAGE="Hello World!" WINDOWTITLE="Message Box
Title" POSITION="lower_left" ICON="information" />
```

Arrays

Like variables, arrays are used to represent data in a task that may be different each time a task runs. But unlike standard variables, arrays can contain multiple rows and optionally multiple columns. You may want to think of an array as nothing more than a list, such as a shopping list, to-do list, birthday list, etc. All items in an array, like the items in a list, have a position somewhere between first and last. Items are numbered from lowest to highest in arrays, therefore, they are accessed by number. For example, to retrieve the element in Row 2, Column 10, the following expression would be entered:

```
%arrayName(2,10)%
```

One & Two Dimensional Arrays

An array can be used to store a list of data read from a text file or other system containing customer data. For instance, if there are 10 customers, the array would need to have 10 rows. If the data consisted of first name, last name, and company name it would need 3 columns and would need to be a two dimensional array. An array can also be a simple list of text or numeric values (e.g., multiple rows of data, only one column), this is called a one dimensional array.

An array with only one dimension is linear. In other words, it contains a list of data that can be referenced by a number. For example if my one dimensional array named myArray had 3 values, then the syntax would be:

```
myArray(1) = value1
myArray(2) = value2
myArray(3) = value3
```

In a two dimensional array, you have both rows and columns, such as a spreadsheet. You would then need to reference a cell by row and column, like the expression below which references row 1, column 5.

```
%Myarray(1,5)%
```

Creating & Setting Arrays

The Array action contains individual activities that allow you to create or modify an array. To create an array, use the Create array activity. In the properties of this activity, you give your array a name, choose whether you want your array to be a one, two, or three dimensional array and optionally, set its values. After creating an array, you can set its values in any task step using the Set array activity or you can re-size an array using the Resize array activity.

Examples

Sample 1

This is a simple task that associates each array with a color.

```
<AMARRAY NAME="myarray" TYPE="TEXT" ROWS="3" />
<AMSET VARIABLENAME="myarray(1)">Red</AMSET>
<AMSET VARIABLENAME="myarray(2)">Blue</AMSET>
<AMSET VARIABLENAME="myarray(3)">Green</AMSET>
<AMSHOWDIALOG>%myarray(1)%
%myarray(2)%
%myarray(3)%</AMSHOWDIALOG>
```

Sample 2

This is a more complex task that uses functions and performs loops (requires a folder called C:\test\ with a few files in it).

```
<AMVARIABLE NAME="thefilename"></AMVARIABLE>
<AMVARIABLE NAME="counter"></AMVARIABLE>
<AMARRAY NAME="myarray" TYPE="TEXT" ROWS="%FileCount('c:\test\')%"
/>
<AMLOOP TYPE="FOLDER" FOLDER="c:\test\"
RESULTVARIABLE="thefilename">
<AMINCREMENTVARIABLE RESULTVARIABLE="counter" />
<AMSET VARIABLENAME="myarray(%counter%)">%thefilename%</AMSET>
</AMLOOP>
```

```
<AMLOOP TOTALLOOPS="%UBound(myarray, 1)%"
RESULTVARIABLE="counter">
<AMSHOWDIALOG>%myarray(counter)%</AMSHOWDIALOG>
</AMLOOP>
```

What are Constants?

Like variables, constants are used to identify data in a task. However, unlike a variable, a constant is an identifier whose associated value is "fixed," therefore, it cannot typically be altered by a task during its execution. The elements of a constant consists of a name and value. A constant's defined value is global, thus, are available to all tasks on a particular system. When you modify a constant's value, every occurrence of that value is updated across the system.

Constants are useful for defining values that are used often but may change over time or modifying multiple instances of a value at one time. For example, you could create a constant named `AdminEmail` and assign the network administrator's email address as the value. You could use this constant in multiple tasks where you wanted to send an e-mail notification to the network administrator. If you wanted to run such a task at another location with a different network administrator, you would set up a constant of the same name at the other location and assign it whatever value you wanted. If the network administrator's e-mail address changed, you would simply change the value of the constant.

Using Constants

Constants may be used in any step parameter by specifying the name of the constant surrounded by percent (%) signs. For example, if a constant named `BackupFolder` was initially created and given the value of an existing directory used to normally backup files, specify this directory as the destination in a Copy File action, and enter `%BackupFolder%` in the **Destination** parameter of this action.

To add a constant in an action using the Expression Builder

1. In the Properties dialog of an action, click inside any entry box then click **Insert Expression/Variable** which appears next to any entry box that accepts expressions.
2. In the Expression Builder window that appears, select the folder labeled **Constants** from the list of folders located in the bottom left pane.

3. Double-click a selection from the list of the defined constants for the local machine that are populated in the bottom right pane. This populates the top pane with the selected item. The selected item will be automatically surrounded with percent signs upon insertion, therefore, there is no need to surround the item with percent signs at this point.
4. Click **Insert** to properly insert the constant and close the Expression Builder window. For further information, refer to "Using the Expression Builder" in the Automate Desktop help.

Datasets

What are Datasets?

In the simplest terms, a dataset is any named group of records. Like variables, datasets are used to represent data in a task that may be different each time a task runs. But unlike standard variables, datasets can contain multiple rows and columns. This is useful when retrieving a collection of data, such as a database or spreadsheet, or retrieving information that describes one or more objects or items. Datasets can hold information such as medical or financial records. Datasets are also used to store information returned by applications or the operating system itself. For example, Advanced Workflows returns information about task errors (e.g., error name, number, description) via an AMError dataset. Datasets can be cataloged, which permits them to be referred to by name without specifying where they are stored.

The data in a dataset is laid out like a database table, which has a unique name and consists of columns and rows. The columns consist of pre-defined units of data, such as one item in a database or personnel data about one member of a customer list. The rows contain the actual data for the columns. An example of a simple dataset containing customer data is illustrated below. The name of the dataset is Customers. The first row (in bold) contains the unique names of the dataset columns, which in this case, describes the data type. All other rows include the actual data as described by each column.

Table: Customers

First	Last	Email	Phone
John	Brown	John.Brown@mydomain.com	626 222-2222
Steven	Goldfish	goldfish@fishhere.net	323 455-4545
Paula	Smith	ps@mycompany.org	416 323-8888
James	May	jim@supergig.co.uk	416 323-3232

Using Datasets

Datasets are accessed in the same way that one would access information in a database, by specifying the column and row where the data resides. Every dataset created and used must have a unique name, much like variables. But because datasets are more like tables (as opposed to variables, which can be thought of more like containers that hold one value), they are referenced differently than other objects. When using datasets, the unique name of the dataset must be referenced followed by the column name enclosed in percentage signs. For instance:

```
%DatasetName.ColumnName%
```

When a dataset is created, the current row is automatically set to 1 (assuming that the dataset has any data, since it is possible for a dataset to have 0 rows, such as when a SQL Query returns no data). A dataset is of minimal use, however, unless one can access the other rows. Typically, this is accomplished by using the Loop Dataset action which takes a dataset name as a parameter and automatically increments the current row with each iteration. The loop continues until all the rows have been accessed. In this way, one could make a task that performs operations on each row of the dataset while using the same expression. Using the table above as an example, one can retrieve the email address of each customer using the following expression inside a Loop Dataset action:

%Customers.Email%

Using a Loop Dataset step is not the only way to access dataset rows. It is possible to directly access a particular row of a dataset by supplying the row number within the expression. For example, if the dataset contains five rows and you need to get the data in row 2, simply enter the row number enclosed in parenthesis directly after the dataset name. For example:

%DatasetName(2).ColumnName%

Again, using the table illustrated above, if you want to retrieve the phone number specified in row 3 (in this case, Paula Smith's phone number), the following expression will do the trick:

%Customers(3).Phone%

Common Actions that use Datasets

Several actions and activities create and populate datasets. The table below describes some of those actions.

Action Name	Description
SQL Query	Queries a database and populates a dataset with the data retrieved.
Get Email	Retrieves one or more messages from the specified server and populates a dataset with the results.
SNMP Get	Populates a dataset with the data sent by the agent.

Pre-Named Datasets

In addition to datasets that are created and populated during execution of an action or activity, a collection of pre-named (or fixed field) datasets are also available which can provide more insight about the behavior of a running task, determine system states or examine other elements, such as triggers used to fire off a task or errors that were generated by a task. The table below describes some of these types of datasets.

Dataset Name	Description
AMError	Determine specific characteristics about the task error that occurred, including error code or number, error description, action/activity that generated the error, and other values.

Example using SQL Query action

The SQL Query action is an example of an action that creates and populates a dataset. The fields contained within that dataset are determined by the query that was executed. For example if the following query is executed:

```
SELECT firstname, lastname,
company from customer where
city='Los Angeles';
```

Then the following dataset would be generated:

datasetname
|--firstname
|--lastname
|--company

A record (row) is created for each record (row) that is retrieved from the server. To access this data use the Loop Dataset action to loop through the records. Inside the loop you can extract the data from the field of your choice (from the current record) by using an embedded expression such as the one that follows:

```
%mydatasetname.firstname%
```

or you could combine two fields together like this:

```
%mydatasetname.firstname + ' ' +
mydatasetname.lastname%
```

Expressions such as these can be used in any parameter in any action. The AML code to display the data in a message box would look like this:

```
<AMMESSAGEBOX MESSAGETEXT="%mydatasetname.firstname%"
WINDOWTITLE="The firstname of the current record is">
```

At runtime the text %mydatasetname.firstname% is replaced by the contents of the subject of the current record.

The percent signs (%) at the beginning and end of the variable name indicates that the text in-between the percent signs is an expression and should not be taken literally. Instead, it is replaced with the current contents of that column in the current row at runtime.

Common Dataset Fields

Most of the fields (columns) that are returned in a dataset are dictated by the action creating and populating the dataset. For example, when using the SQL Query action, the field names are controlled by the columns returned by the query. In the SNMP Get action, the field names are dependant on the SNMP query data being returned by the agent. Nonetheless, a common set of fields are available which can be used to retrieve information about any created dataset. Most of these fields are global, which means they can be used on any dataset, regardless of which action or activity the dataset originated from. These fields can be accessed in the same manner as other fields, though some of their values are read only. Also, the values of these fields are the same regardless of the row being accessed.

The table below describes the common set of fields (columns) that a dataset creates (assuming the dataset name assigned is theDataset).

Name	Data Type	Return Value
theDataset.CurrentRow	Number	The current row that will be accessed in the dataset by an expression that does not contain a specific row index.
theDataset.TotalRows	Number	The total number of rows in the dataset
theDataset.TotalColumns	Number	The total number of columns (not including the static columns) in the dataset.
theDataset.ExecutionDate	Date	The date and time the dataset was created and populated
theDataset.RowsAffected	Number	The number of rows affected by an update.
theDataset.SQLQuery	Text	The SQL Query that was used to generate this dataset (If a SQL Query was not used, this value is empty).
theDataset.Datasource	Text	The data source used for the SQL Query, if applicable.

Name	Data Type	Return Value
theDataset.ColumnNames	Text	A comma-delimited list of the column names in the dataset

Expressions

An expression, in programming, is a combination of symbols that represents a value. The expression is interpreted according to the rules of the programming language and a value is returned. The Expression Builder provides a quick and convenient way to create and evaluate BASIC expressions from directly within the parameters of a task step. Expressions work by taking the text found between percentage (%) signs and passing it to the BASIC expression interpreter. During runtime, the BASIC interpreter replaces the original expression (including the percentage signs) and returns the expression's resulting value instead.

Using Expressions

Many actions that normally require expressions may already be built into activities. However, you can use expressions to further expand an action's capabilities. For instance, you can build expressions that resolve complex, dynamic data at runtime with the use of variables, functions, extended functions and operators.

Example 1 - Simple mathematical expression

In the following example, a task runs a single step that performs a simple mathematical calculation and displays the results in a message box. To create this task:

1. In the Task Builder, add a Message Dialog activity to the Steps panel.
2. From the General properties of this activity, enter %7+7% in the Message to display parameter (as shown below) and click OK to save changes.
3. From the Task Builder's ribbon, click the Run button to start the task.

Since the phrase 7+7 is surrounded in percentage signs, knows that it should try to resolve the expression and not display it literally. To resolve the expression, at runtime, the value between the percentage signs is passed to the BASIC expression interpreter, where it is processed and the result is returned. A message dialog should appear with the value of 14 (as shown below).

Example 2 - Expression with variable

Expressions can also contain variables. Taking the previous example a step further, assume we now want the task to give the user the ability to enter a number he/she would like squared.

1. Use the Create variable activity to create a variable named NUMTOSQUARE with the initial value of 1. This variable will be used during runtime to retain the user's response to the question.
2. Use the Input dialog activity as the second step to display the question What number would you like squared? and populate the NUMTOSQUAREvariable with the answer.
3. Use a Message Dialog as the last step which will display the result by use of an expression multiplying the variable by itself. %NUMTOSQUARE *.

The complete task is displayed below in AML format. For convenience, the AML code can simply be copied and pasted directly into the Steps panel of the Task Builder.

```
<!-- Create the variable. -->
<AMVARIABLE NAME=
<!-- Ask the user to enter a number. -->
<AMINPUTBOX
<!-- Perform the calculation using an inline expression. -->
<AMMESSAGEBOX%>
```

Notes

Notice that we first had to create the variable we intend to use. All variables to be passed into a scripting language, either explicitly through the use of a BASIC script step or implicitly through the use an expression, need to be created first. You can accomplish this using the Create Variable action.

Variable names are not case sensitive; the variables are capitalized here for clarity.

Because each expression is passed to the scripting language engine, any command available from a common BASIC script is available to you for use in expressions in your step parameters. Refer to BASIC Scripting for details on these powerful commands.

Functions and Extended Functions

In computer programming, a function is predefined code which can generate various kinds of values depending on certain input from the user. Most programming languages such as VBScript come with a pre-written, built-in range of functions that perform various procedures or routines. Functions can be used as an expression in any task step, thus, adding more intelligence to a task and further broadening the functionality. Additionally, contains an extensive collection of additional functions that increase the functionality of the Scripting Engine which are called extended functions.

Function Syntax

Most functions require other parameters to properly complete a procedure or routine. The general format of a function is its name followed by any arguments contained in between parenthesis. An argument (sometimes referred to as parameters) is a value sent to the function when it is called upon. By passing an argument to a function, the function is given information to work on. A basic function would look like this:

```
FunctionName(arguments)
```

Certain functions may not need arguments to properly perform its duty. A function without arguments must include an empty set of parentheses (). For example, the Date() function returns the current system date and the Now() function returns the current date along with the time. These functions require no arguments or parameters in which they need to perform any procedures on, therefore the proper syntax are simply the functions themselves, with nothing entered in between the parenthesis. For instance, if the following syntax was entered:

```
Date()
```

The returned result would be the current date value such as 4/1/2024.

An example of a function which requires arguments is the Len() function, which returns the number of characters in a string. The syntax for this function is Len(“string”) which requires a string to be entered inside the parenthesis in which it will perform calculations on. For example, if this was entered:

```
Len(“Hello”)
```

The returned value would be 5.

Another example is the extended function, ExtractFileName(), which requires a path and filename in between the parenthesis in which it will extract the file name from. So, if the following was entered:

```
ExtractFileName(c:\
```

The returned value would be filename.txt.

An example of a function requiring more elaborate arguments is the Left() function, which returns a specified number of characters from the left side of a string. The syntax for this function is Left(“string”, length) which requires a valid string followed by the amount of characters to return inside the parenthesis. If the following was entered:

```
Left("AutoMate", 4)
```

The return value would be Auto.

Another example is the InsStr()function which returns the position of the first occurrence of one string within another. The search begins at the first character of the string. The syntax for this function is InStr(“string1”,“String2”). So, if the following was entered:

```
InStr(Hello,o)
```

The return value would be 5 signifying that the letter “o” is the fifth character in the word “Hello”.

> **NOTE:** Sample syntax for each function and extended functions can be found in the Help document.

Using Functions

Built-in VBScript functions along with extended functions can be used to return data inside any action parameter of a task that accepts expressions by surrounding the function with percent signs. This tells that the contents in between the percent signs should not be taken literally, but used as an expression instead. For example, the Len() function can be used within Message dialog activity by simply entering the following syntax in the Message to display parameter:

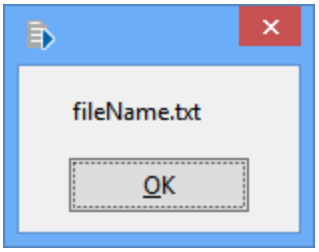
```
%Len("Hello")%
```

As previously mentioned, the Len() function returns the number of characters in a string. When the task runs, the Len() function will be performed and a message dialog displaying the results will appear, as illustrated below.

Expressions such as variables, constants or other functions can be used as arguments or parameters entered inside the parenthesis of a function. In such cases, the specified expression needs to be entered by itself (omitting any percent signs, quotes or other characters). For example, assume that a variable named %theFile% is populated with the string value c:\folderName\fileName.txt. to extract only the filename from this string and view the results in a message dialog, the proper syntax to enter in a Message Dialog step would be:

```
%ExtractFileName(theFile)%
```

During task execution, a message dialog will display the properly extracted filename as shown below.



Using Expression Builder

The Expression Builder is a valuable tool used to assist in the creation and examination of expressions. When using the Expression Builder, help regarding each function can be accessed by first selecting the Functions folder from the lower left pane, then selecting the desired function from the lower right pane and pressing the F1 key or by right-clicking the function and selecting Help from the pop-up menu that appears.