

# Studio User Guide

RELEASE: 2023.4 ▾

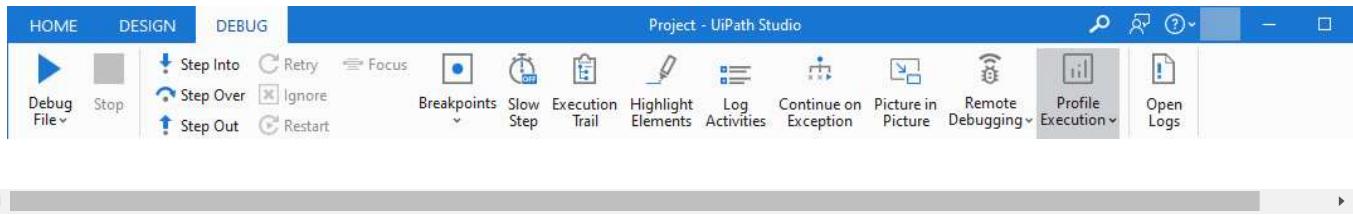
## TABLE OF CONTENTS

### [Debugging Actions](#)

- [Step Into](#)
- [Step Over](#)
- [Step Out](#)
- [Retry](#)
- [Ignore](#)
- [Restart](#)
- [Break](#)
- [Focus](#)
- [Slow Step](#)
- [Execution Trail](#)
- [Highlight Elements](#)
- [Log Activities](#)
- [Continue on Exception](#)
- [Picture in Picture](#)
- [Remote Debugging](#)
- [Profile Execution](#)
- [Open Logs](#)

## Debugging Actions

Debugging of a single file or the whole project can be performed both from the **Design** or **Debug** ribbon tabs. However, the debugging process is not available if project files have validation errors.



## Step Into

Use **Step Into** to debug activities one at a time. When this action is triggered, the debugger opens and highlights the activity before it is executed.

When **Step Into** is used with **Invoke Workflow File** activities, the workflow is opened in a new tab in **ReadOnly** mode and each activity is executed one by one.

The keyboard shortcut for **Step Into** is **F11**.

## Step Over

Unlike the **Step Into** action, **Step Over** does not open the current container. When used, the action debugs the next activity, highlighting containers (such as flowcharts, sequences, or **Invoke Workflow File** activities) without opening them.

This action comes in handy for skipping analysis of large containers which are unlikely to trigger any issues during execution.

**Step Over** is available using the **F10** keyboard shortcut.

## Step Out

As the name suggests, this action is used for stepping out and pausing the execution at the level of the current container. **Step Out** completes the execution of activities in the current container, before pausing the debugging. This option works well with nested sequences.

**Step Out** is available using the **Shift + F11** keyboard shortcut.

## Retry

**Retry** re-executes the previous activity, and throws the exception if it's encountered again. The activity which threw the exception is highlighted and details about the error are shown in the **Locals** and **Call Stack** panels.

## Ignore

The **Ignore** action can be used to ignore an encountered exception and continue the execution from the next activity so that the rest of the workflow can be debugged.

This action is useful when jumping over the activity that threw the exception and continuing debugging the remaining part of the project.

## Restart

**Restart** is available after an exception was thrown and the debug process is paused. The action is used for restarting the debugging process from the first activity of the project. Use **Slow Step** to slow down the debugging speed and properly inspect activities as they are executed.

Please take into consideration that when using this option after using the **Run from this Activity** action, the debugging is restarted from the previously indicated activity.

## Break

**Break** allows you to pause the debugging process at any given moment. The activity which is being debugged remains highlighted when paused. Once this happens, you can choose to **Continue**, **Step Into**, **Step Over**, or **Stop** the debugging process.

It is recommended to use **Break** along with **Slow Step** so that you know exactly when debugging needs to be paused.

An alternative to using **Slow Step** in this situation is to keep an eye on the **Output** panel and use **Break** on the activity that is currently being debugged.

## Focus

**Focus Execution Point** helps you return to the current breakpoint or the activity that caused an error during debugging. The **Focus** button is used after navigating through the process, as an easy way to return to the activity that caused the error and resume the debugging process.

Alternatively, when debugging is paused because a breakpoint was reached, **Focus** can be used for returning to said breakpoint, after navigating through activities contained in the automation process.

A third case is when the debugging is paused either after using **Step Into** or **Step Over** and then navigating through the process. In this case, **Focus** returns to the activity that paused the debugging process.

From the **Breakpoints** context menu, you can select **Focus** to highlight the activity with the breakpoint.

## Slow Step

**Slow Step** enables you to take a closer look at any activity during debugging. While this action is enabled, activities are highlighted in the debugging process. Moreover, containers such as flowcharts, sequences, or **Invoke Workflow File** activities are opened. This is similar to using **Step Into**, but without having to pause the debugging process.

**Slow Step** can be activated both before or during the debugging process. Activating the action does not pause debugging.

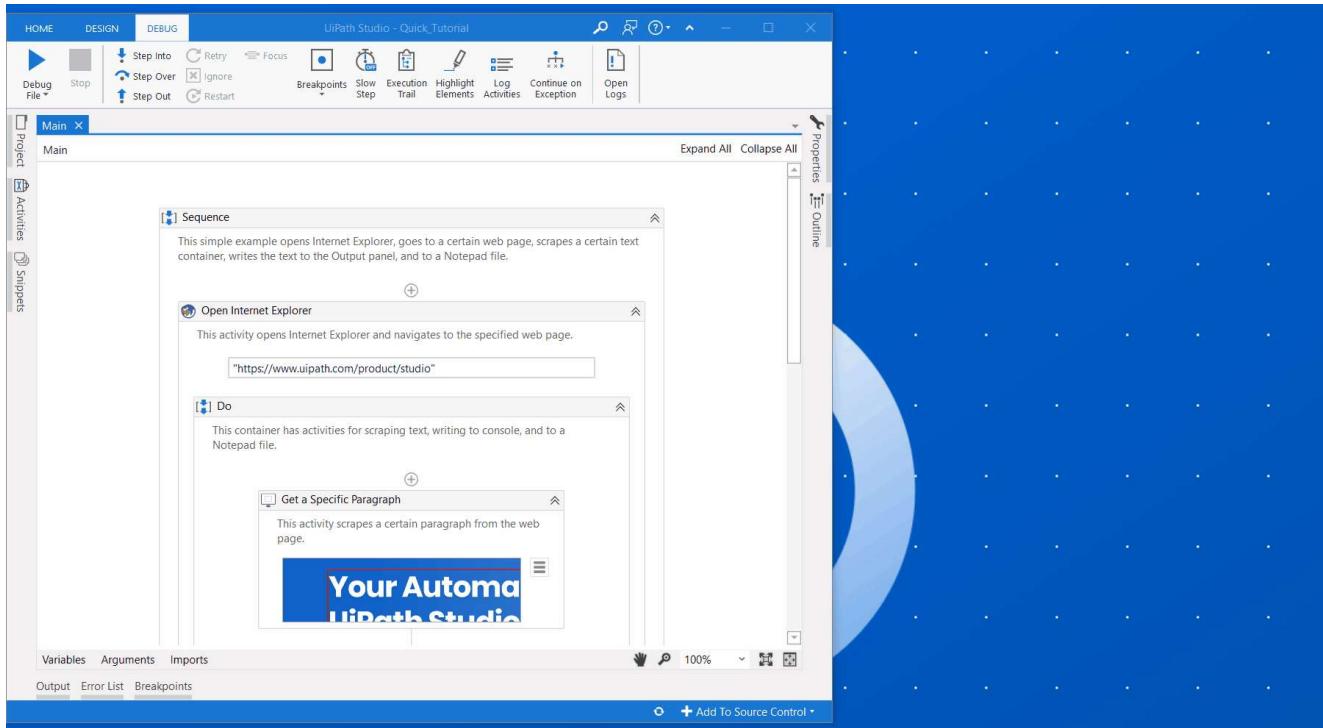
Although called **Slow Step**, the action comes with 4 different speeds. The selected speed step runs the debugging process slower than the previous one. For example, debugging with **Slow Step** at 1x runs it the slowest, and fastest at 4x. In other words, the speed dictates how fast the debugger jumps from one activity to the next.

Each time you click **Slow Step** the speed changes by one step. You can easily tell by the icon, which updates accordingly.

## Execution Trail

The **Execution Trail** ribbon button is disabled by default. When enabled, it shows the exact execution path at debugging. As the process is executed, each activity is highlighted and marked in the **Designer** panel, showing you the execution as it happens:

- executed activities are marked and highlighted in green;
- activities that were not executed are not marked in any way;
- activities that threw an exception are marked and highlighted in red.



## Highlight Elements

If enabled, UI elements are highlighted during debugging. The option can be used both with regular and step-by-step debugging.

## Log Activities

If enabled, debugged activities are displayed as Trace logs in the **Output** panel. Note that **Highlight Elements** and **Log Activities** options can only be toggled before debugging, and persist when reopening the automation project. This is not applicable for invoked workflows unless these files are opened in the **Designer** panel.

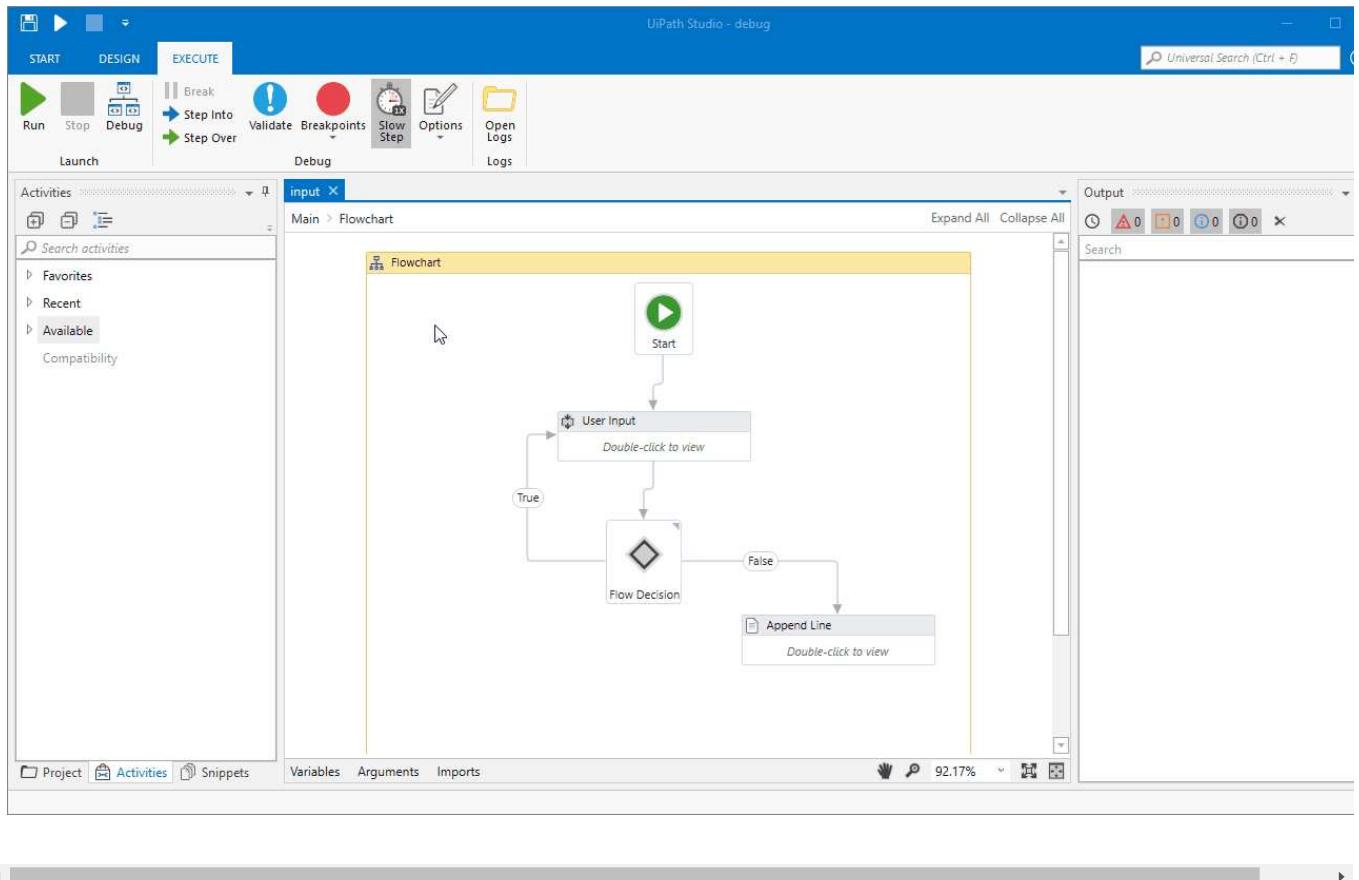
Logs are automatically sent to Orchestrator if connected, but you can have them stored locally by disabling the **Allow Development Logging** option from the **Robot Settings** tab in the Add or Edit user window.

Disabling **Log Activities** can be a way to send smaller log files to Orchestrator.

### **NOTE:**

When running processes from Studio, the logs sent to Orchestrator are always Trace when Log Activities is disabled, and Verbose when Log Activities is enabled. This overrides both the Robot and the Orchestrator setting.

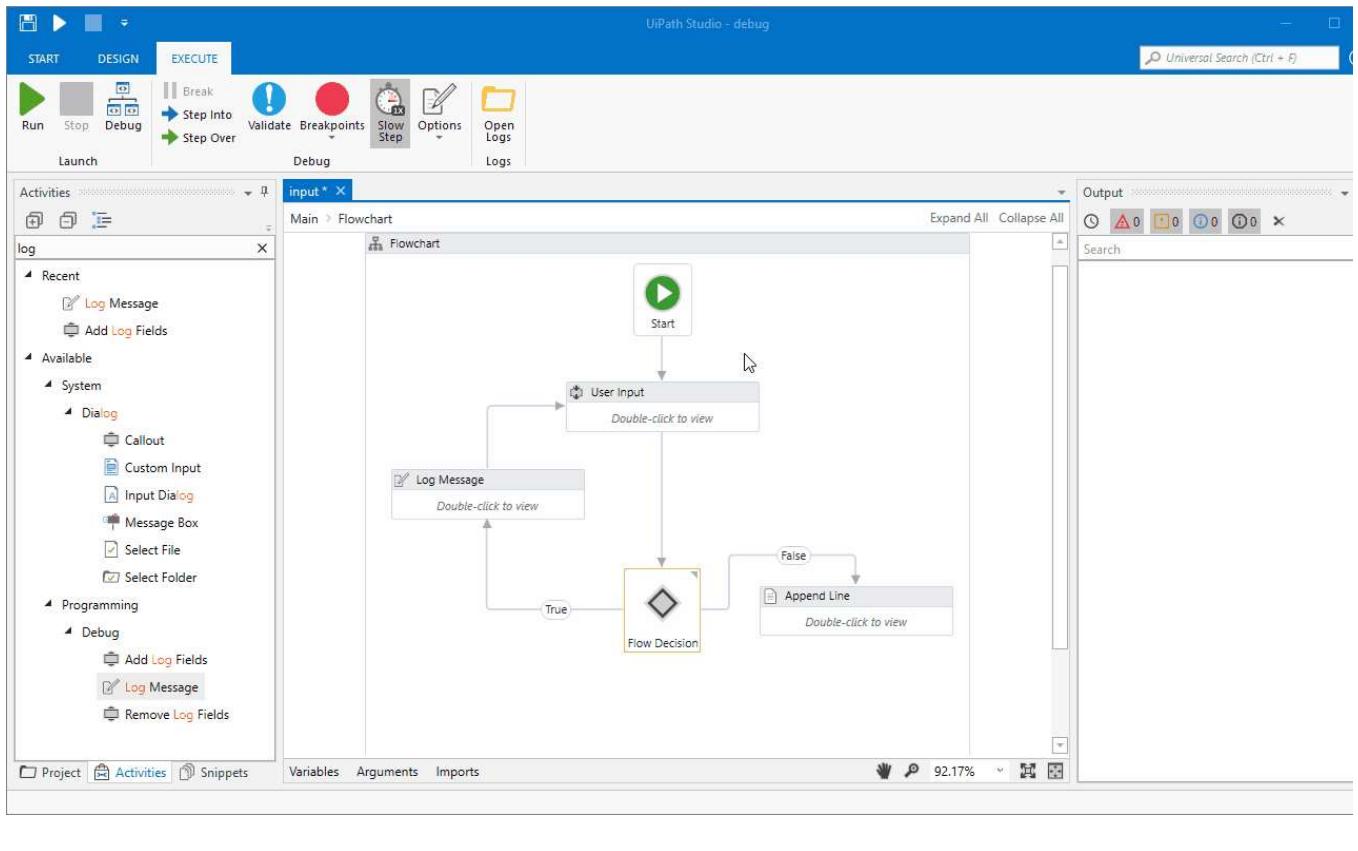
By default, the debugger logs activities so that each step appears in the **Output** panel. We recommend leaving it enabled for easier tracing, as you can see in the image below:



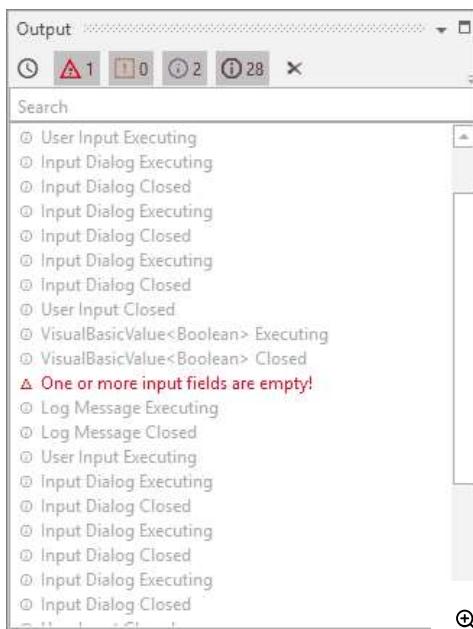
The issue here is that one or more input fields from the User Input sequence are blank, which is a True condition for the Flow Decision. You can tell this by the fact that, during debugging, the User Input sequence is executed twice, meaning that one or more fields were left blank during the first execution.

If you decide to disable the **Log Activities** option for debugging, Trace logs are not displayed in the **Output** panel. In the case of a normal execution with no errors, you only get to see the debug execution start and end times. However, adding a **Log Message** can help you determine where issues might occur.

For example, you can add a **Log Message** activity to inform you that, in this case, one or more input fields are empty. This message appears in the **Output** panel during debugging, even if the **Log Activities** option is disabled, as you can see below:



Remember that you can always filter the messages displayed in the [Output panel](#) by simply selecting the alert types of interest, or even clear all messages.



Note that by default all debugging logs are sent to Orchestrator. You can disable this by clearing the **Allow Development Logging** option from the [Settings tab](#) in the Add or Edit Robot window. If this option is disabled, debugging logs are only stored locally.

## Continue on Exception

This debugging feature is disabled by default. When disabled in the ribbon, it throws the execution error and stops the debugging, highlights the activity which threw the exception, and logs the exception in the [Output panel](#). If a [Global Exception Handler](#) was previously set in the

project, the exception is passed on to the handler.

When enabled, the exception is logged in the **Output** panel, the execution continues.

## Picture in Picture

The **Picture in Picture** ribbon option in the **Debug** tab is available for both executing and debugging processes or libraries in a separate session on your machine.

If enabled, whenever you select **Run** or **Run File**, **Debug** or **Debug File** the process starts either in a separate session or in a virtual desktop in the user session. If **Picture in Picture** is disabled, debugging and execution is performed in the current session.

Having the option to run a process in Picture in Picture (PiP) can be very useful in attended automation. Verify whether a process runs successfully in PiP, and then update the project settings to indicate if it can be executed using this feature after it is published:

1. In the **Project** panel, click **Settings**  to open the Project Settings window.

2. In the **General** tab:

- **PiP Options** - Indicate whether the project was tested using Picture in Picture and whether it should start in PiP by default.
  - **Tested for PiP usage; Starting in PiP** - The automation has been approved to run in PiP mode. When run, it starts in PiP by default.
  - **Tested for PiP usage; Not starting in PiP by default** - The automation has been approved to run in PiP mode. When run, it starts in the main session or desktop by default.
  - **Not tested for PiP usage** - The automation has not been approved to run in PiP mode. When run, it starts in the main session or desktop by default. If run in PiP, a dialog informs the user it was not tested using this feature and prompts for confirmation before proceeding.
- **PiP Type** - Select how to isolate the automation from the user session when running the project in PiP: **New Session** (child session on the machine) or **New Desktop** (virtual desktop in the user session).

For more information, including limitations of this feature, see [Picture in Picture](#) in the Robot Guide.

## Remote Debugging

When this feature is enabled, all run and debug operations are performed on a specified remote robot instead of the robot installed locally, allowing you to test the automation on different environments. For more information, see [Remote Debugging](#).

## Profile Execution

You can identify performance bottlenecks in the workflow when you debug the file. For more information, see [Profile Execution](#).

## Open Logs

Clicking **Open Logs** brings up the %localappdata%\UiPath\Logs folder where logs are locally stored. The naming format of log files is YYYY-DD-MM\_Component.log (such as 2018-09-12\_Execution.log, or 2018-09-12\_Studio.log). Read more about logging [here](#).



Default Theme

English

# Studio User Guide

RELEASE:

2023.4



## TABLE OF CONTENTS

### [Logging Levels](#)

[Log Message Types](#)[Default Logs](#)[User-Defined Logs](#)[Log Fields](#)[Default fields](#)[Type-specific fields](#)[User-defined fields](#)

# Logging Levels

The following table lists the logging levels in UiPath.

Logging Level	Default Logs	User-Defined Logs
Off	None	None
Critical	All messages logged with Critical level or higher.	All messages logged with Critical level or higher.
Error	All messages logged with Error level or higher.	All messages logged with Error level or higher.
Warning	All messages logged with Warning or higher.	All messages logged with Warning or higher.
Information	All messages logged with Information or higher.	All messages logged with Information or higher.
Trace	All messages logged with Trace level or higher.	All messages logged with Trace level or higher.
Verbose	All messages logged with Trace level and Workflow Tracking logs.	All messages logged with Trace level.

The **Verbose** level logs a message for both the activity **start** and **end**, plus the values of the variables and arguments that are used.

By default, the Verbose level includes:

- **Execution Started** log entry - generated every time a process is started.
- **Execution Ended** log entry - generated every time a process is finalized.
- **Transaction Started** log entry - generated every time a transaction item is obtained by the robot from Orchestrator.
- **Transaction Ended** log entry - generated every time the robot sets the transaction status to either Success or Failed.
- **Activity Information** log entry - generated every time an activity is **started**, **faulted** or **finished** inside a workflow.

 **NOTE:**

The priority order of the log types is: **Verbose < Trace < Information < Warning < Error <**

**Critical < Off.**

Log Level	Logged	Example / Comment
<b>Verbose</b>	<b>Activities</b>	`Trace {"message": {"DisplayName": "Message box", "State": "Executing", "Activity": "UiPath.Dialog.Activities.MessageBox"}, "Caption": "", "Text": "String in message BOX"}
<b>Verbose</b>	<b>Variables</b>	`"Variables": {"NewTransaction": "False"}
<b>Verbose</b>	<b>Arguments (properties)</b>	`"Arguments": {"Caption": "", "Text": "String in message BOX", "Ch
<b>Trace</b>	<b>Activities</b>	`Trace {"message": {"DisplayName": "Main", "State": "Executing", "Activity": "System.Activities.Statements.WriteLine"}}
<b>Information</b>	<b>WriteLine</b> <b>Log Message</b>	Info {"message": "message from activity"}  <b>NOTE:</b> Except messages logged with Trace level set in activity.
<b>Warning</b>	<b>Warnings</b>	Warn {"message": "Warning from log message activity"}
<b>Warning</b>	<b>Errors</b>	Error {"message": "Error from log message activity"}
<b>Warning</b>	<b>Critical</b>	Critical Errors
<b>Error</b>	<b>Errors</b>	Error {"message": "Error from log message activity"}
<b>Error</b>	<b>Critical</b> <b>Fatal</b>	Critical Errors
<b>Critical</b>	<b>Critical</b> <b>Fatal</b>	Critical Errors
<b>OFF</b>	n/a	n/a

# Log Message Types

There are several possible occurrences of log messages, depending on the event that is logged, as follows:

## Default Logs

Generated by default when the execution of a project starts and ends, when a system error occurs and the execution stops, or when the logging settings are configured to log the execution of every activity.

 **NOTE:**

These logs have the **Default** value in the **logType** field.

The events logged by this category are:

- **Execution Start** is generated every time a process is started. This is logged starting with the **Information** logging level.
- **Execution End** is generated every time a process is finalized. This is logged starting with the **Information** logging level.
- **Transaction Start** is generated every time a transaction within a process is started. This is logged starting with the **Information** logging level.
- **Transaction End** is generated every time a transaction within a process is finalized. This is logged starting with the **Information** logging level.
- **Error Log** is generated every time the execution encounters an error and stops. This is logged starting with the **Error** logging level.
- **Debugging Log** is generated if the Robot Logging Setting is set to Verbose and contains, activity names, types, variable values, arguments etc. This is logged starting with the **Trace** logging level.

## User-Defined Logs

Generated according to the process designed by the user in Studio, when using the **Log Message** activity or the **Write Line** activity.

 **NOTE:**

These logs have the **User** value in the **logType** field.

If such logs are generated at an interval lower than 1ms, they may be improperly displayed in the Output panel.

# Log Fields

There are multiple types of log fields that can be found throughout the above log message types. These can be classified as follows:

## Default fields

These log fields are present in all execution type logs, such as SQL (if configured), Elasticsearch (if configured), and the default EventViewer Logs:

- Message - The log message.
- Level - Defines the log severity.
- Timestamp - The exact date and time the action was performed.
- FileName - The name of the .xaml file being executed.
- jobId - The key of the job running the process.
- processName - The name of the process that triggered the logging.
- processVersion - The version number of the process.
- windowsIdentity - The name of the user that performed the action that was logged.
- robotName - The name of the robot (as defined in Orchestrator).

 **NOTE:**

The processName and processVersion fields do not appear in logs if the process is run locally, without being connected to Orchestrator.

## Type-specific fields

These logs are present depending on the log type:

- Execution End
  - totalExecutionTimeInSeconds
  - totalExecutionTime
- Transaction Start
  - queueName

- transactionID
- transactionState
- Transaction End
  - queueName
  - transactionID
  - transactionState
  - transactionStatus
  - transactionExecutionTime
  - processingExceptionType
  - processingExceptionReason
  - queueItemReviewStatus
  - queueItemPriority
- Debugging Log
  - activityInfo, which is a JSON message with the following fields:
    - DisplayName
    - State (Faulted, Closed, Executing)
    - Activity
    - Variables
    - Arguments

 **NOTE:**

Only totalExecutionTimeInSeconds, totalExecutionTime and queueName are always present in the log messages. Variables and Arguments usually have sub-fields.

## User-defined fields

These fields are defined in Studio by using the **Add Log Fields** activity and appear in all subsequent logs after the activity is generated, unless they are removed by the **Remove Log Fields** activity.

 **NOTE:**

Creating user-defined log fields that have the same name as a default log field causes the logging process to become corrupted and may cause issues in the workflow you are running.

For example, creating a user-defined log field called jobId causes this issue, as jobId is a log field that is generated by default.



Default Theme

English



# Robot User Guide

RELEASE:

2023.4



## TABLE OF CONTENTS

### [Robot Logs](#)

[Robot Execution Logs](#)

[Execution Logs Logging Level](#)

[Changing the Default Logging Level from the Assistant:](#)

[Orchestrator target](#)

[NLog target](#)

[Deleting old workflow log files](#)

[Further Editing of Logs](#)

[Robot Diagnostic Logs](#)

[Driver Diagnostic Logs](#)

## Robot Logs

Logs are time-stamped files that contain informational events, error and warning messages relevant to the application.

The types of logs are presented in the following pages:

# Robot Execution Logs

Robot Execution Logs are messages generated by the execution of a process. They contain information related to its behavior and user-defined messages in the workflow.

Execution logs are generated by:

- The **Write Line** activity, which creates logs at the Trace level.
- The **Log Message** activity, which creates logs at the level specified in the **Level** property field of the activity.
- Running an automation project, which generates logs that contain the behavior of each activity. These logs use Trace level if the **Level** setting in the **Orchestrator Settings window** in Assistant is set to **Verbose**.

## Execution Logs Logging Level

The default logging level is controlled by the **Level** setting stored in **Orchestrator Settings window**. By default, it is set to **Information**.

 **NOTE:**

The [log level set in Orchestrator](#) for the robot overrides the one [set in the UiPath Assistant](#).

## Changing the Default Logging Level from the Assistant:

1. From the UiPath Assistant, go to the Preferences menu and then access **Orchestrator Settings**
2. Select the desired logging level from the **Log Level** drop-down menu, under the **Logging** section.

The screenshot shows the 'Preferences' window in the UiPath application. On the left, a sidebar lists 'General', 'Keyboard Shortcuts', 'Orchestrator Settings' (which is selected and highlighted in orange), 'Launchpad', 'Tools', and 'Help'. The main content area is divided into two sections: 'Orchestrator Configuration' and 'Logging'. Under 'Orchestrator Configuration', there is a 'Connection Type' section with a dropdown set to 'Service URL' containing the value 'https://cloud.uipath.com/'. Below it is a 'Service URL' input field also containing 'https://cloud.uipath.com/'. A blue 'Sign out' button is located below these fields. Status information shows 'Status: Connected, Licensed' with a green circular icon. Under 'Logging', there is a 'Log Level' section with a dropdown set to 'Information'. A note below states: 'During execution of your automation, information logs may contain sensitive data. More details can be found in our [Assistant documentation](#).'. The UiPath logo is visible at the bottom left of the window.

**NOTE:**

If the **Robot** is installed in service mode, administrator permissions are needed to edit this setting.

## Orchestrator target

If the **Robot** is connected to **Orchestrator**, all execution logs are sent to **Orchestrator** and can be seen in the [Logs](#) page.

If Orchestrator is unavailable, logs are stored in a local database (`C:\Windows\SysWOW64\config\systemprofile\AppData\Local\UiPath\Logs\execution_log_data`), within the available disk space, until the connection is restored. When the connection is restored, the logs are sent in batches in the order they had been generated.

**ⓘ NOTE:**

The database is not deleted after the logs have been successfully sent to Orchestrator.

## NLog target

Additionally, log targets and content can be configured by editing the `<Installation Folder>\NLog.config` file.

The target location of the logs is controlled by the `<Installation Folder>\NLog.config` file. The Diagnostic logs are collected by the **Internal** type logger and are forwarded by using **NLog targets**. By default, Execution Logs are stored in a file in the `%LocalAppData%\UiPath\Logs` folder. The messages are collected by the **WorkflowLogging** logger and can be forwarded by using **NLog targets**, as specified by the following parameters in the `NLog.config` file:

```
<?xml version="1.0" encoding="utf-8" ?>
<nlog xmlns="http://www.nlog-project.org/schemas/NLog.xsd" xmlns:xsi="http://www.w3.org/200
  <variable name="WorkflowLoggingDirectory" value="${specialfolder:folder=LocalApplicationD
  <rules>
    <logger name="WorkflowLogging" writeTo="WorkflowLogFiles" final="true" />
  </rules>
  <targets>
    <target type="File" name="WorkflowLogFiles" fileName="${WorkflowLoggingDirectory}/#${sho
  </targets>
</nlog>
```

## Deleting old workflow log files

To avoid creating a large amount of log files consuming disk space on the machine, you can choose to archive log files once a specific number files has been reached.

This is done by adding the following parameters in the `<target>` tag `NLog.config`:

- `archiveNumbering="Date"`
- `archiveEvery="Day"`
- `archiveDateFormat="yyyy-MM-dd"`
- `archiveFileName="${WorkflowLoggingDirectory}/{{#}}_Execution.log"`
- `maxArchiveFiles="10"`

```
<?xml version="1.0" encoding="utf-8" ?>
<nlog xmlns="http://www.nlog-project.org/schemas/NLog.xsd" xmlns:xsi="http://www.w3.org/
  <variable name="WorkflowLoggingDirectory" value="${specialfolder:folder=LocalApplicati
  <rules>
    <logger name="WorkflowLogging" writeTo="WorkflowLogFiles" final="true" />
  </rules>
  <targets>
    <target type="File"
      name="WorkflowLogFiles"
      fileName="${WorkflowLoggingDirectory}/${shortdate}_Execution.log"
      layout="${time} ${level} ${message}"
      keepFileOpen="true"
      openFileCacheTimeout="5"
      concurrentWrites="true"
      encoding="utf-8"
      writeBom="true"
      archiveNumbering="Date"
      archiveEvery="Day"
      archiveDateFormat="yyyy-MM-dd"
      archiveFileName="${WorkflowLoggingDirectory}/{#}_Execution.log"
      maxArchiveFiles="10"
    />
  </targets>
</nlog>
```

**ⓘ NOTE:**

The `maxArchiveFiles` is the parameter controlling the number of archive files.

**ⓘ NOTE:**

Editing the `NLog.config` file requires administrator permissions. The robot service does not need to be restarted for the changes to take effect.

## Further Editing of Logs

If the logging level is set to **Verbose**, the messages contain all the details about the activities that were run at execution. This log output can be customized by editing the `UiPath.Executor.exe.config` file, from `C:\Program Files\UiPath\Studio` folder.

For this, the following XML code must be added under the `<system.serviceModel>` tag.

```
<tracking>
  <profiles>
    <trackingProfile name="StandardProfile">
      <workflow>
        <activityStateQueries>
          <activityStateQuery activityName="*">
            <states>
              <state name="Faulted"/>
            </states>
            <arguments>
              <argument name="*"/>
            </arguments>
            <variables>
              <variable name="*"/>
            </variables>
          </activityStateQuery>
        </activityStateQueries>
      </workflow>
    </trackingProfile>
  </profiles>
</tracking>
```

Since the `<states>` tag contains only `<state name="Faulted"/>`, inserting the above code enables only the activities which have the **Faulted** state to be logged. Adding other parameters under the `<states>` tag, such as `<state name="Executing"/>` causes activities that have other states to be logged as well.

Not only activity states can be modified, but also variables and arguments. More information about customization can be found [here](#).

 **NOTE:**

Modifying the `UiPath.Executor.exe.config` file requires a [restart of the robot service](#) for changes to take effect.

## Robot Diagnostic Logs

The robot diagnostic logs provide information related to the **Robot** itself and its context. They are useful to identify the cause of a specific error.

Robot diagnostic logs are saved in the following locations:

- On Windows, diagnostic logs related to errors are logged in the **Event Viewer**.

- On all platforms, including Windows, verbose diagnostic logs are written to a file:
  - %LocalAppData%\UiPath\Logs\internal\Robot.log on Windows when the robot is installed in user mode.
  - %PROGRAMDATA%\UiPath\Logs\internal\Robot.log on Windows when the robot is installed in service mode.
  - ~/.local/share/UiPath/Logs/internal/Robot.log on other platforms.

## Driver Diagnostic Logs

**Driver Tracing** is written to an .etl file.

**To enable Driver Tracing**, open Command Prompt with administrator rights, access the installation directory using the cd argument, such as cd C:\Program Files\UiPath\Studio, and run the UiRobot.exe --enableLowLevel command.

**To disable Driver Tracing**, open Command Prompt with administrator rights, access the installation directory using the cd argument, such as cd C:\Program Files\UiPath\Studio, and run the UiRobot.exe --disableLowLevel command.

### NOTE:

We recommend enabling Low Level Tracing only while investigating a problem, and disabling it when the investigation session is over.

Additionally, the .etl file containing the trace information is generated only after you disable the feature.

The Robot does not have to be restarted for the changes to take effect.



Default Theme

English

# Orchestrator Standalone User Guide

DELIVERY:

AUTOMATION CLOUD

AUTOMATION SUITE

STANDALONE

RELEASE:

2023.4



## TABLE OF CONTENTS

### [Orchestrator Logs](#)

[Orchestrator Diagnostic Logs](#)

[Enabling UiPath Orchestrator Diagnostic Logs](#)

[Targets of the UiPath Orchestrator Diagnostic Logs](#)

[Orchestrator Execution Logs](#)

[Targets of the Orchestrator Execution Logs](#)

[Elasticsearch Server](#)

[X-PACK Authentication](#)

# Orchestrator Logs

# Orchestrator Diagnostic Logs

These are diagnostic logs generated by UiPath Orchestrator regarding its behavior.

## Enabling UiPath Orchestrator Diagnostic Logs

The UiPath Orchestrator Diagnostic Logs are enabled once UiPath Orchestrator is successfully installed. They rely on the NLog infrastructure and their configuration can be found in the `UiPath.Orchestrator.dll.config` file, under the `<nlog>` tag.

 **NOTE:**

Please keep in mind that both server exceptions from Orchestrator, and the stack trace on the **Job Details** window, are logged in English, regardless of what language was chosen by the user.

## Targets of the UiPath Orchestrator Diagnostic Logs

All application logs are logged to the Event Viewer at the minimum logging level of **Information**. This is specified by the following lines in the `UiPath.Orchestrator.dll.config` file:

```
<target xsi:type="EventLog" name="eventLog" layout="${message}" source="Orchestrator"
log="Application" />
<logger name="*" minlevel="Info" writeTo="eventLog" />
```

Logs generated by the **Jobs Scheduler** have a separate target and logger:

```
<target xsi:type="EventLog" name="eventLogQuartz" layout="[Quartz] ${message} ${onexception:
${exception:format=tostring}}" source="Orchestrator" log="Application" />
<logger name="Orchestrator.Quartz.*" minlevel="Info" writeTo="eventLogQuartz" />
```

Example:

- Could not create Quartz Job

Logs generated by **business and other validation rules** have a separate target and logger:

```
<target xsi:type="EventLog" name="businessExceptionEventLog"
layout="${message}${onexception:${exception:format=tostring:maxInnerExceptionLevel=5:innerFormat=tostring}}" source="Orchestrator.BusinessException" log="Application" />
<logger name="BusinessException.*" minlevel="Info" writeTo="businessExceptionEventLog"
final="true" />
```

These types of error messages are logged in the Event Viewer in the following cases:

- validation issues such as:
  - Invalid username/email address or password.
  - The machine name DOC is already taken
- business conflicts such as:
  - License expired!
  - The floating robot's session is already active on machine ROQADOC06!
  - The robots already have pending jobs for this Process.
- not found exceptions such as:
  - QueueName1 does not exist.

## Orchestrator Execution Logs

The Orchestrator Execution Logs are sent by the **Robots** connected to it and are displayed in the **Logs** section of the **Jobs** or **Robots** pages. The application receives the data from the Robots, adds its own parameters (**TenantID**, **FolderID**), and forwards the messages to different targets, as specified in the `<nlog>` section from the `UiPath.Orchestrator.dll.config` file.

## Targets of the Orchestrator Execution Logs

By default, all Robot logs are sent to the **Logs** table of the Default Orchestrator Database, where UiPath Orchestrator stores other information, but the `UiPath.Orchestrator.dll.config` file can be configured to send them to a different Database as well.

The **Logs** page displays information from the **Logs** table of the **Default Database**. So, if this section does not exist, or the logs are saved to a different database, the page is empty. All parameters should be according to the table schema, which looks like this:

```

<target xsi:type="Database" name="database" connectionString="${ui-connection-strings:item=database}">
  <commandText>
    insert into dbo.Logs (OrganizationUnitId, TenantId, TimeStamp, Level, WindowsIdentity, ProcessName)
    values (@organizationUnitId, @tenantId, @timeStamp, @level, @windowsIdentity, @processName)
  </commandText>
  <parameter name="@organizationUnitId" layout="${event-properties:item=organizationUnitId}" />
  <parameter name="@tenantId" layout="${event-properties:item=tenantId}" />
  <parameter name="@timeStamp" layout="${date}" />
  <parameter name="@level" layout="${event-properties:item=levelOrdinal}" />
  <parameter name="@windowsIdentity" layout="${event-properties:item=windowsIdentity}" />
  <parameter name="@processName" layout="${event-properties:item=processName}" />

```

```
<parameter name="@jobId" layout="${event-properties:item=jobId}" />
<parameter name="@message" layout="${message}" />
<parameter name="@rawMessage" layout="${event-properties:item=rawMessage}" />
</target>
<logger name="Robot.*" writeTo="database" final="true" />
```

Other targets can be added to the logs by configuring the `UiPath.Orchestrator.dll.config` file. A list of available targets can be found [here](#).

 **NOTE:**

When upgrading Orchestrator, Nlog targets are deleted and recreated, as follows:

- Upon upgrade to 2022.4, Nlog database targets are reverted to their default values.
- Upon upgrade to 2022.10, Nlog database targets are deleted and replaced with new and improved targets.

This covers `database`, `monitoring`, and `insightsRobotLogs` targets.

 **IMPORTANT:**

If the number of Robot logs stored in the table is higher than 1 million, we recommend creating the following index for improved search performance:

```
CREATE NONCLUSTERED INDEX [IX_Search] ON [dbo].[Logs]
(
    [TenantId] ASC,
    [OrganizationUnitId] ASC,
    [Level] ASC,
    [TimeStamp] DESC
)WITH (STATISTICS_NORECOMPUTE = OFF, DROP_EXISTING = OFF, ONLINE = OFF, OPTIMIZE_FOR_SEQUEN
GO
```

## Elasticsearch Server

By default, there's an Elasticsearch target configured from the installation script. The index is different for each tenant, but this can be configured from the specified target in the `<nlog>` section.

### For Elasticsearch versions lower than 8.0:

```
<target name="robotElasticBuffer" xsi:type="BufferingWrapper" flushTimeout="5000">
<target xsi:type="ElasticSearch" name="robotElastic" uri="uritoelasticsearchnode" index="${
</target>
</target>
```

## For Elasticsearch versions 8.0 and higher:

```
<target name="robotElasticBuffer" xsi:type="BufferingWrapper" flushTimeout="5000">
<target xsi:type="ElasticSearch" name="robotElastic" uri="uritoelasticsearchnode" index="${
</target>
</target>
```

In order for Elasticsearch versions 8.0 and higher to work properly, these parameters are set as follows:

- `documentType` is empty.
- `enableApiVersioningHeader` is set to true.

# X-PACK Authentication

### NOTE:

By default, the Elasticsearch security features are disabled if you have a basic or trial license. We strongly recommend that you enable them.

## Username-and-password Authentication

To enable authentication via a username and password, you need to take the following steps:

### 1. Configure the Elasticsearch server as follows:

- Add the `xpack.security.enabled` setting to the `elasticsearch.yml` configuration file.
- Set up a username and password.

For more details on this, see the [Elasticsearch documentation](#).

### 2. Configure Orchestrator's `UiPath.Orchestrator.dll.config` file as follows:

- **Option 1:** If you do not use an NLog target, you need to configure the following parameters: `Logs.Elasticsearch.Username` and `Logs.Elasticsearch.Password`. Make sure their values match the

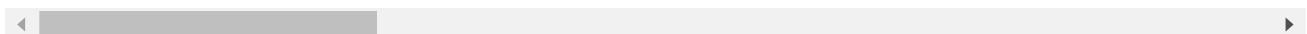
Elasticsearch settings from Step 1.

- **Option 2:** If `Logs.RobotLogs.ReadTarget` is set to an NLog target (for example, `robotElasticBuffer`), and the `Logs.Elasticsearch.Nodes` setting is not specified, configure the target by adding the following: `requireAuth="true" username="XPACKuser" password="p@$$w0rd"`. Make sure these parameter values match the Elasticsearch settings from Step 1.

For more on these parameters, see the [UiPath.Orchestrator.dll.config](#) page.

For a configuration example, see the following:

```
<target name="robotElasticBuffer" xsi:type="BufferingWrapper" flushTimeout="5000">
    <target xsi:type="ElasticSearch" name="robotElastic" uri="" requireAuth="true" u
    </target>
```



- **Option 3:** If `Logs.RobotLogs.ReadTarget` is set to an NLog target ( for example, `robotElasticBuffer`), and an Elasticsearch node is specified via the `Logs.Elasticsearch.Nodes` setting, you need to explicitly configure it (as it overrides the NLog target settings), and make sure to also add the following: `requireAuth="true" username="XPACKuser" password="p@$$w0rd"`. Make sure these parameter values match the Elasticsearch settings from Step 1.

## OAuth2 Authentication

To switch to OAuth2 as an authentication method for Elasticsearch, you need to take the following steps. Note that you need to provide your current credentials to switch to this token-based authentication method.

1. Configure the Elasticsearch server as follows:

- Enable TLS (HTTPS) for the transport layer.
  - Update the following settings in the `elasticsearch.yml` configuration file:
    - `xpack.security.authc.token.enabled: true`
    - `xpack.security.enabled: true`
    - `xpack.security.authc.token.timeout` - This setting is optional and controls for how long a token is valid. By default, its value is set to 20 minutes.
- For more details on this, see the [Elasticsearch documentation](#).

2. Update the following parameters in Orchestrator's `UiPath.Orchestrator.dll.config` file to reflect the settings you opted for at Step 1.

- `Logs.Elasticsearch.TlsEnabled = "true"` - By default, this parameter is set to true and ensures TLC (HTTPS) is enabled.
- `Logs.Elasticsearch.OAuthEnabled = "true"` - By default, this parameter is set to false. For more on this, see [Logs.Elasticsearch.OAuthEnabled](#).

- `Logs.Elasticsearch.OAuthExpireInSeconds = "1200"` - This parameter is optional unless the default value of 1200 is changed in the Elasticsearch `xpack.security.authc.token.timeout` setting. This parameter must have the same value as in the Elasticsearch configuration. For more on this, see [Logs.Elasticsearch.OAuthExpireInSeconds](#).

 **NOTE:**

The first two steps help you configure a token-based authentication mechanism for reading logs. If you use NLog, an additional step is needed.

3. To enable OAuth2 for NLog, make sure to also configure the following parameter in Orchestrator's `UiPath.Orchestrator.dll.config` file. Note that you must fill in the [username and password](#) for authentication in Elasticsearch because the initial token is generated based on those credentials.

- `OAuthEnabled = "true"` - By default, it is set to false. For more on this, see the [UiPath.Orchestrator.dll.config](#) page.

 **IMPORTANT:**

If `Logs.RobotLogs.ReadTarget` is set to an NLog target (for example, `robotElasticBuffer`), and the `Logs.Elasticsearch.Nodes` setting is not specified, then the `Logs.Elasticsearch.OAuthEnabled` is filled from the NLog target configuration. Same logic is applied for username and password.

## API key authentication

To enable authentication via API key, follow the steps outlined below.

1. Generate the API key by following [these steps](#).
2. [Store](#) the API key as a secret in your Azure key vault.
3. Configure the following NLog target parameters with your data, thus creating a connection between Orchestrator and your key vault which allows the key to be retrieved:

```
apiKeyEnabled="true"
apiKeyProvider="AzureKeyVault"
apiKeySecretName=""
azureKeyVaultUri=""
azureKeyVaultDirectoryId=""
azureKeyVaultClientId=""
azureKeyVaultCertificateThumbprint=""
azureKeyVaultCertificateStoreLocation="CurrentUser/LocalMachine"
```

The following parameters need to be edited with your values:

- <SecretName> - the name you set for your API key in the key vault
- <KeyVaultUri> - the URI of your key vault
- <KeyVaultDirectoryId> - your key vault directory ID
- <KeyVaultClientId> - your key vault client ID
- <KeyVaultCertificateThumbprint> - the thumbprint of your key vault certificate
- CurrentUser/LocalMachine - the location where the certificate is stored

## API key expiration

By default, API keys do not expire, but you can still choose to set an expiration date for them.

If your API key is set to expire, you must generate a new and store it in the key vault before the expiration date, so as to make sure that Orchestrator can always retrieve a valid key.

Orchestrator reads API keys from the key vault every 15 minutes, so this is the maximum amount of delay you can expect before your new key is propagated.



Default Theme

English

# Orchestrator Standalone User Guide

DELIVERY: AUTOMATION CLOUD AUTOMATION SUITE STANDALONERELEASE: 2023.4 

## TABLE OF CONTENTS

### [About Queues and Transactions](#)

[Queues Overview](#)[Schema Definitions](#)[Transactions Overview](#)[Processing Order](#)[Exporting Transactions](#)[Queue SLA Predictions](#)[Risk SLA](#)[Queue SLA Permissions](#)

## About Queues and Transactions

A queue is a container that enables you to hold an unlimited number of items. Queue items can store multiple types of data, such as invoice information or customer details. This information can be processed in other systems – SAP or Salesforce, for instance.

The data stored in, and output from, Queue items is free form by default. For situations where a specific schema is needed, such as integrations with other applications, processing of machine generated forms, or for analytics, you can upload custom [JSON schemas](#) to ensure that all Queue item data is in the proper format.

In Orchestrator, newly created queues are empty by default. To populate them with items you can either use the upload functionality in Orchestrator, or Studio activities. The latter also enable you to change item statuses and process them. As soon as queue items are processed, they become transactions.

## Queues Overview

Queues enable you to create large automation projects underlined by complex logic. For example, you can create a process that collects all invoice information and creates a queue item for each piece of data to store it. Subsequently, you can create another process that gathers the information from Orchestrator, and uses it to perform additional tasks, such as paying the invoices in a different application, postponing their payment according to their due date or value, sending emails to the accounting team every time a bill is paid, etc.

The **Queues** page enables you to create new queues. It also provides you with viewing access on previously created queues, charts with the transaction status progress over time, and on various other details, such as average execution time and the total number of successful transactions.

The screenshot shows the 'Queues' tab selected in the top navigation bar. The main area displays a table of queue items with the following columns: NAME, DESCRIPTION, IN PROGRESS, REMAINING, AVERAGE TIME, SUCCESSFUL, APP EXCEPT..., BIZ EXCEPT..., and PROCESS. The table contains three rows:

NAME	DESCRIPTION	IN PROGRESS	REMAINING	AVERAGE TIME	SUCCESSFUL	APP EXCEPT...	BIZ EXCEPT...	PROCESS
DocQueue		0	264	0 seconds	0	0	0	<span>⋮</span>
ReportsQueue		0	434	0 seconds	0	0	0	<span>⋮</span>
Test_Queue		0	0	0 seconds	0	0	0	<span>⋮</span>

At the bottom right of the table, there are pagination controls: 'Items 10' and '1-3 / 3'. Below the table is a horizontal scrollbar.

Item statuses are controlled by RPA developers when they create the automation projects, while revision statuses are controlled in Orchestrator and enable you to perform version control, but **only of queue items that have been abandoned or have failed with an application or business exception**.

Failed or abandoned items can also be assigned to a reviewer, which can be changed or cleared at any point, if needed. Each of these changes are tracked in the **History** tab of the **Audit Details** window. The reviewer is in charge of assessing the current status of the transactions he is assigned to, and changing the review status. The status of queue items up for revision can be changed in the **Review Requests** page.

## Schema Definitions

When creating or editing a queue, you can upload a custom JSON schema for the **Specific Data**, **Output Data**, and/or **Analytics Data**. With the schema(s) in place all transactions are validated against the provided format, and if the resulting data does not conform that item fails with a Business Exception.

### ⚠️ IMPORTANT:

- The schema is not applied retroactively to existing transactions, only to those executed after you have uploaded the schema(s).
- Your schema(s) must **not contain an array**.
- For validation purposes, `DateTime` is accepted as `string` type.
- Use of and validation of an **Analytics Data** schema requires Robots and Activities of version 19.10 or greater.
- If the uploaded schema(s) do not contain a valid schema definition URI, then the `draft-07` definition, as in the below example, is used as the fallback.

### ⓘ NOTE:

To achieve better control in terms of Orchestrator performance, the **Specific Data** size of queue items is limited to 1 MB with the help of the `Queue.MaxSpecificDataSizeInKiloBytes` app setting. Anything beyond this limit cannot be added to a queue, and it returns the 403 - Payload Too Large error code. If you need to upload larger items, store the large data in an external storage and only reference the link in the item.

A sample schema:

json

```
{
  "definitions": {}}
```

```

"$schema": "http://json-schema.org/draft-07/schema#",
"$id": "http://example.com/root.json",
"type": "object",
"title": "The Root Schema",
"additionalProperties": { "type": "string" },
|required": [
  "stringTest",
  "intTest",
  "boolTest"
],
"properties": {
  "stringTest": {
    "$id": "#/properties/stringTest",
    "type": "string",
    "title": "The Stringtest Schema",
    "default": "",
    "examples": [
      "stringTest"
    ],
    "pattern": "^(.*)$"
  },
  "intTest": {
    "$id": "#/properties/intTest",
    "type": "integer",
    "title": "The Inttest Schema",
    "default": 0,
    "examples": [
      30
    ]
  },
  "boolTest": {
    "$id": "#/properties/boolTest",
    "type": "boolean",
    "title": "The Booltest Schema",
    "default": false,
    "examples": [
      false
    ]
  }
}
}

```

## Transactions Overview

The **Transactions** page displays the transactions from a given queue. It also shows their statuses, the dates when they should be processed, the Robot that processed them, and the type of exception thrown or assigned reference, if any.

You can search for a specific transaction or a group of them, according to a custom reference, which is added through the **Reference** property of the **Add Queue Item** and **Add Transaction Item** activities. The reference can be used to link your transactions to other applications used within an automation project. Additionally, this feature enables you to search for certain transactions in Orchestrator, according to the provided custom reference.

Transaction references can also be enforced to be unique, at queue level. This feature is enabled when creating the queue and applies to all transactions except deleted or retried ones. This makes identifying a specific item a breeze and eases the review process.

If a duplicate reference is encountered while adding items to a queue, the job fails with a Faulted status and displays the **Execution error: UiPath.Core.Activities.OrchestratorHttpException: Error creating Transaction. Duplicate Reference.** error message in the **Job Details** window.

Information stored in queue items is displayed in Orchestrator, in the **Transaction Details** window, under **Specific Data**. Additionally, if the item failed and was retried, the history of the item is displayed in the same window.

STATUS	REVISION	DEADLI...	RETRY ...	POSTPO...	STARTED	ENDED	RO...	EXCEPT...
Failed	None	10/29/201...	1		a day ago	a day ago	G	Application
Retried	None	10/29/201...	0		a day ago	a day ago	G	Application

Items 10 ▾ 1 - 10 / 0 | < < > >

CLC

The **Transaction Details** window contains three tabs:

- **Details** - enables you to view the exact information added to a transaction, as well as the statuses it went through, and the number of times it was retried.
- **Comments** - enables you to view and add transaction-related comments in case you need to share information about a specific transaction with your teammates. All the users with **View**, **Edit**, and **Delete** permissions on Queues and Transactions can add, edit, or remove comments, respectively, however, keep in mind you can make changes to your own comments only.
- **History** - enables you to see what action was performed by who, see who the reviewer is and what the review status is.

## Processing Order

Within any given queue the transactions are processed in a hierarchical manner, according to this order:

1. Items that **have a Deadline**, as follows:
  - in order of **Priority**; and
  - according to the set **Deadline** for items with the same **Priority**.
2. Items with **no Deadline**, in order of **Priority**, and
  - according to the rule **First In, First Out** for items with the same **Priority**.

When setting a **Deadline** or a Postpone date, we recommend populating the respective fields with relative dates. For example, `DateTime.Now.AddHours(2)`, `DateTime.Now.AddDays(10)` and `DateTime.Now.Add(New System.TimeSpan(5, 0, 0, 0))`. Additionally, you can use the US notation to add an exact time, such as `10/10/2019 07:40:00`. Automatic correction of this date is available, for example, if you write `12 10 2019 9:0`, it is automatically transformed to `12/10/2019 09:00:00`.

The dates added in Studio for the **Deadline** and **Postpone** fields are displayed in Orchestrator, in the **Transactions** page, under the **Deadline** and **Postpone** columns.

## Exporting Transactions

You can export all the transactions and information related to a given queue to a .csv file, by clicking the **Export** button, in the **Transactions** page. All page filtering options apply to the generated file, too.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	Status	Revision	Reference	Exception	Deadline	Priority	Robot	Postpone	Started	Ended	Transaction	ExRetry No.	Specific Data	Key	Reviewer Name	Exception Reason	Output
41	Deleted	None		BusinessException	Normal			#####	#####	#####	#####	0	0 ("DynamicProp 0856f3d2-0fb1-411b-9035-f7cc OnUSCheck amount is bigger				
42	Deleted	None		BusinessException	Normal			#####	#####	#####	#####	0	0 ("DynamicProp a66eb053-231d-49ae-87d0-8ct OnUSCheck amount is bigger				
43	Deleted	None		BusinessException	Normal			#####	#####	#####	#####	0	0 ("DynamicProp a784d713-b10c-48d4-823b-ca5 OnUSCheck amount is bigger				
44	Deleted	None		BusinessException	Normal			#####	#####	#####	#####	0	0 ("DynamicProp 9cf53093-52e4-4fc-9133-05fe OnUSCheck amount is bigger				
45	Deleted	None		BusinessException	Normal			#####	#####	#####	#####	0	0 ("DynamicProp Prop 5bd350c2-3efc-4be5-biae-ea6 OnUSCheck amount is bigger				
46	Deleted	None		BusinessException	Normal			#####	#####	#####	#####	0	0 ("DynamicProp 2e4cb928-bb49-4518-bfe6-d57 OnUSCheck amount is bigger				
47	Deleted	None		BusinessException	Normal			#####	#####	#####	#####	0	0 ("DynamicProp 047fe81-95b7-42ab-a33b-38 OnUSCheck amount is bigger				
48	Deleted	None		BusinessException	Normal			#####	#####	#####	#####	0	0 ("DynamicProp Prop 047fe81-95b7-42ab-a33b-38 OnUSCheck amount is bigger				
49	Deleted	None		BusinessException	Normal			#####	#####	#####	#####	0	0 ("DynamicProp 0c863d3f-a438-4eac-8736-dc0 OnUSCheck amount is bigger				
50	Deleted	None		BusinessException	Normal			#####	#####	#####	#####	0	0 ("DynamicProp Prop 091fa91-16f2-4cd-b032-0e21 OnUSCheck amount is bigger				
51	Deleted	None		BusinessException	Normal			#####	#####	#####	#####	0	0 ("DynamicProp a3d750a3-7641-4fad-812a-e02 OnUSCheck amount is bigger				
52	Deleted	None		BusinessException	Normal			#####	#####	#####	#####	0	0 ("DynamicProp 21aecaef-8c54-452c-b098-28e OnUSCheck amount is bigger				

To ensure the best performance, please note that the exported entries are not in reverse chronological order.

## Queue SLA Predictions

This tool helps you set an SLA (item Deadline) for newly added items in a queue. This helps you assess if they can be processed in a timely manner, and what resources you need to allocate such that their SLA is not breached. Whenever the SLA is in danger of not being met, you are properly notified such that you can make adjustments accordingly.

The SLA only applies to those items that don't have a deadline set, meaning that a newly added item with no deadline defined beforehand, has it automatically filled in according to the value set as the SLA. Specifically, each item's deadline is represented by the value set for the queue SLA from the moment the item was added into the queue. For instance, if I set the SLA to 2 hours, and I add 3 items into the queue at 4, 5, and 6 PM, then my items have the deadlines 6, 7, 8 PM, respectively.

Items which have a Deadline (either set in Studio or in a .csv file used for upload) are **not** affected by the SLA setting.

### **IMPORTANT:**

- The Priority of the items added in a queue after enabling SLA predictions is automatically set to **High**, regardless of how it was set in Studio or in the .csv file used for upload.
- You cannot delete a process associated to a queue with enabled SLA predictions.
- If at least one queue item exceeds its deadline, **Over Capacity** is displayed in the Necessary Robots (SLA) column and predictions are no longer calculated.
- Predictions are made for queue items with deadlines in the next 24 hours (can be changed using the `Queue.SlaReadaheadTimeLimitHours`), and do not take into account the items' defer dates.

Queue triggers and SLA predictions are interdependent in terms of queue-process association. So whenever configuring one, the other is prefilled such as to have parity between the configurations. Say I define a queue trigger for queue Y to use process X. SLA predictions for queue Y can only be made using process X, therefore X is prefilled and read-only when enabling queue SLA for Y.

## Risk SLA

You can also define a Risk SLA for your items, which works like a buffer zone before the SLA. Explicitly, the risk deadlines of your items are calculated based on the Risk SLA from the moment the queue item was added in the queue. Say I set the Risk SLA to 2 hours, and I add 3

items into the queue at 4:30, 5:15, and 6:45 PM, then my items have the risk deadlines 6:30, 7:15, 8:45 PM, respectively.

After the Risk SLA has passed and the queue item is not processed, the item becomes at risk of not meeting its deadline. The user is properly notified, such that he can make adjustments accordingly.

## Queue SLA Permissions

In order to be able to configure SLA predictions for a queue you to be granted the following permissions:

- **View on Processes**
- **View on Queues**
- **Edit on Queues** (to configure SLA when editing a queue)
- **Create on Queues** (to configure SLA when creating a queue)



Default Theme

English



Home

Orchestrator

Field Descriptions for the Test Data Queues Page - Standalone  
2023.4

# Orchestrator Standalone User Guide

DELIVERY:

AUTOMATION CLOUD

AUTOMATION SUITE

STANDALONE

RELEASE:

2023.4

## TABLE OF CONTENTS

### [Field Descriptions for the Test Data Queues Page](#)

[The Test Data Queues Page](#)

[The Add Test Data Queue Window](#)

[Test Data Queue Items](#)

# Field Descriptions for the Test Data Queues Page

## The Test Data Queues Page

Field	Description
<b>Search</b>	Search for test data queues that match your input, according to the name and description.
<b>Add</b> 	Open the <b>Add Test Data Queue</b> window, to create new test data queues.
<b>Name</b>	The name of the test data queue.
<b>Items Count</b>	The number of items included in a test data queue.
<b>Consumed Items Count</b>	The number of items that have been used. Items flagged as <b>Consumed</b> will be excluded from future test data queue runs.
<b>Created</b>	The time when the test data queue was created.
<b>Last Modified</b>	The time when the test data queue was last edited.
<b>More Actions</b>	Open a menu with the following options: <ul style="list-style-type: none"> <li>• <a href="#">Edit</a></li> <li>• <a href="#">View Items</a></li> <li>• <a href="#">Upload Items</a></li> <li>• <a href="#">Delete</a></li> </ul>

## The Add Test Data Queue Window

Field	Description
<b>Name</b>	A custom name for the test data queue to help you identify it.
<b>Description</b>	A custom description for the test data queue. Consider adding a description to help you easily identify the use of each particular test data queue.

Field	Description
<b>Content JSON Schema</b>	A form that lists your uploaded JSON schema. It includes a reference on <a href="#">How to create your own JSON schema</a> .
<b>Browse</b>	A button that opens a search window to find and upload your JSON file. You can only open JSON file types.
<b>Cancel</b>	Close the window without adding a test data queue and return to the previous window.
<b>Add</b>	Save your configuration and create your test data queue.

## Test Data Queue Items

Field	Description
<b>ID</b>	The ID of the record as part of the JSON schema.
<b>USERID</b>	The user ID of the record as part of the JSON schema.
<b>Is-Consumed</b>	The items that have been used are flagged as <b>Consumed</b> . The consumed items will be excluded from future test data queue runs.



Default Theme

English

# Workflow Activities

## TABLE OF CONTENTS

### [Add Queue Item And Get Reference](#)

Description

Project compatibility

Windows, Windows - Legacy configuration

Cross-platform configuration

## Add Queue Item And Get Reference

`UiPath.Persistence.Activities.Queue.AddQueueItemAndGetReference`

## Description

Adds an Orchestrator Queue Item with parameters and retrieves the corresponding Queue Item Object, stored in a `QueueItemData` variable, in order to wait for transaction completion at any time during the execution of the workflow.

## Project compatibility

Windows - Legacy | Windows | Cross-platform

# Windows, Windows - Legacy configuration

## Designer panel

- **Queue Name** - The queue where the QueueItem object is to be added. The name is case insensitive, meaning that if in Orchestrator it was defined as "MyFirstQueue", it matches "myfirstqueue". The maximum number of characters is 50.
- **Priority** - The priority level of the Queue Item that is added. This property is a criterion for the prioritization of queue items, alongside **Due Date** and **Defer Date**.
- **ItemInformation** - A collection of additional information about the specific QueueItem that is to be added. The information is stored in the item and is used during a transaction processing. It is recommended to use only primitive values of the following types: Number, Boolean, String and DateTime. The value of string arguments cannot contain the following characters: [ and " ". Argument names cannot contain the following characters: :, ., , , @, ".
- **Queue Item (Output)** - The queue object that is returned from Orchestrator as a QueueItemData object after the queue is created. This object can be passed to the [Wait for Queue Item and Resume](#) activity in order to suspend the execution of the workflow until the transaction is completed.

## Properties panel

### Common

- **ContinueOnError** - Specifies if the automation should continue even when the activity throws an error. This field only supports Boolean values (True, False). The default value is False. As a result, if the field is blank and an error is thrown, the execution of the project stops. If the value is set to True, the execution of the project continues regardless of any error.

 **NOTE:**

If this activity is included in **Try Catch** and the value of the **ContinueOnError** property is True, no error is caught when the project is executed.

- **DisplayName** - The display name of the activity.
- **TimeoutMS** - Specifies the amount of time (in milliseconds) to wait for the activity to run before an error is thrown. The default value is 30000 milliseconds (30 seconds).

### Input

- **Deadline** - The timestamp before which the queue item should be processed. This property can be filled in with relative timestamps such as `DateTime.Now.AddHours(2)`, `DateTime.Now.AddDays(10)` and `DateTime.Now.Add(New System.TimeSpan(5, 0, 0, 0))`. Additionally, you can use the US notation to add an exact time, such as `12/10/2017 07:40:00`. Automatically correcting this date is

available. For example, if you write 8 9 2018 9:0, it is automatically transformed to 08/09/2018 09:00:00.

- **Orchestrator Folder Path** - The [path to an Orchestrator Folder](#) different to the one the current process operates in, where you want to create the queue item. To read more on Orchestrator Folders, go [here](#). For **Classic** folders, this property can only be used with **Floating Robots** and only if the current user has the correct task privilege in the target folder. For **Modern** folders, **folder path overriding is not supported**. This field supports only strings and String variables.

#### NOTE:

The **FolderPath** property must be used **only** if the queue item must be created or queried from a folder different than the current one. If the user performing this action does not have the required permissions for the target folder, queue item creation fails and throws a critical error. You can view more info on Folder Permissions [here](#).

- **ItemInformation** - A collection of additional information about the specific QueueItem that is to be added. The information is stored in the item and is used during a transaction processing. It is recommended to use only primitive values of the following types: Number, Boolean, String and DateTime. The value of string arguments cannot contain the following characters: [ and " ". Argument names cannot contain the following characters: :, ., , , @, ".
- **ItemInformationCollection** - Enables importing an entire dictionary of information for a queue item. This field accepts Dictionary<string, object> variables only.
- **Postpone** - The timestamp after which the queue item may be processed. This property is a criterion for the prioritization of queue items, alongside **Priority** and **Due Date**. This property can be filled in with relative timestamps such as `DateTime.Now.AddHours(2)`, `DateTime.Now.AddDays(10)` and `DateTime.Now.Add(New System.TimeSpan(5, 0, 0, 0))`. Additionally, you can use the US notation to add an exact time, such as `12/10/2017 07:40:00`. Automatically correcting this date is available. For example, if you write 8 9 2018 9:0, it is automatically transformed to 08/09/2018 09:00:00.
- **Priority** - The priority level of the Queue Item that is added. This property is a criterion for the prioritization of queue items, alongside **Due Date** and **Defer Date**.
- **Queue Name** - The queue where the QueueItem object is to be added. The name is case insensitive, meaning that if in Orchestrator it was defined as "MyFirstQueue", it matches "myfirstqueue". The maximum number of characters is 50.
- **Reference** - The reference of the Queue Item that is added. The reference can be used to link your transactions to other applications used within an automation project. Additionally, this feature enables you to search for certain transactions, in Orchestrator, according to the provided string.

## Misc

- **Private** - If selected, the values of variables and arguments are no longer logged at Verbose level.

## Output

- **Queue Item (Output)** - The queue object that is returned from Orchestrator as a QueueItemData object after the queue is created. This object can be passed to the [Wait for Queue Item and Resume](#) activity in order to suspend the execution of the workflow until the transaction is completed.

# Cross-platform configuration

- **Queue Name** - The queue where the QueueItem object is to be added. The name is case insensitive, meaning that if in Orchestrator it was defined as "MyFirstQueue", it matches "myfirstqueue". The maximum number of characters is 50.
- **Priority** - The priority level of the Queue Item that is added. This property is a criterion for the prioritization of queue items, alongside **Due Date** and **Defer Date**.
- **ItemInformation** - A collection of additional information about the specific QueueItem that is to be added. The information is stored in the item and is used during a transaction processing. It is recommended to use only primitive values of the following types: Number, Boolean, String and DateTime. The value of string arguments cannot contain the following characters: [ and " ". Argument names cannot contain the following characters: :, ., , , @, ".
- **Queue Item (Output)** - The queue object that is returned from Orchestrator as a QueueItemData object after the queue is created. This object can be passed to the [Wait for Queue Item and Resume](#) activity in order to suspend the execution of the workflow until the transaction is completed.

## Advanced options

### Common

- **ContinueOnError** - Specifies if the automation should continue even when the activity throws an error. This field only supports Boolean values (True, False). The default value is False. As a result, if the field is blank and an error is thrown, the execution of the project stops. If the value is set to True, the execution of the project continues regardless of any error.

 **NOTE:**

If this activity is included in **Try Catch** and the value of the **ContinueOnError** property is True, no error is caught when the project is executed.

- **DisplayName** - The display name of the activity.
- **TimeoutMS** - Specifies the amount of time (in milliseconds) to wait for the activity to run before an error is thrown. The default value is 30000 milliseconds (30 seconds).

### Input

- **Deadline** - The timestamp before which the queue item should be processed. This property can be filled in with relative timestamps such as `DateTime.Now.AddHours(2)`, `DateTime.Now.AddDays(10)` and `DateTime.Now.Add(New System.TimeSpan(5, 0, 0, 0))`. Additionally, you can use the US

notation to add an exact time, such as 12/10/2017 07:40:00. Automatically correcting this date is available. For example, if you write 8 9 2018 9:0, it is automatically transformed to 08/09/2018 09:00:00.

- **OrchestratorFolderPath** - The [path to an Orchestrator Folder](#) different to the one the current process operates in, where you want to create the queue item. To read more on Orchestrator Folders, go [here](#). For **Classic** folders, this property can only be used with **Floating Robots** and only if the current user has the correct task privilege in the target folder. For **Modern** folders, **folder path overriding is not supported**. This field supports only strings and String variables.

 **NOTE:**

The **FolderPath** property must be used **only** if the queue item must be created or queried from a folder different than the current one. If the user performing this action does not have the required permissions for the target folder, queue item creation fails and throws a critical error. You can view more info on Folder Permissions [here](#).

- **ItemInformationCollection** - Enables importing an entire dictionary of information for a queue item. This field accepts Dictionary<string, object> variables only.
- **Postpone** - The timestamp after which the queue item may be processed. This property is a criterion for the prioritization of queue items, alongside **Priority** and **Due Date**. This property can be filled in with relative timestamps such as `DateTime.Now.AddHours(2)`, `DateTime.Now.AddDays(10)` and `DateTime.Now.Add(New System.TimeSpan(5, 0, 0, 0))`. Additionally, you can use the US notation to add an exact time, such as 12/10/2017 07:40:00. Automatically correcting this date is available. For example, if you write 8 9 2018 9:0, it is automatically transformed to 08/09/2018 09:00:00.
- **Priority** - The priority level of the Queue Item that is added. This property is a criterion for prioritizing queue items, alongside **Due Date** and **Defer Date**.
- **Queue Name** - The queue where the QueueItem object is to be added. The name is case insensitive, meaning that if in Orchestrator it was defined as "MyFirstQueue", it matches "myfirstqueue". The maximum number of characters is 50.
- **Reference** - The reference of the Queue Item that is added. The reference can be used to link your transactions to other applications used within an automation project. Additionally, this feature enables you to search for certain transactions, in Orchestrator, according to the provided string.



Default Theme

English

# Orchestrator Standalone User Guide

DELIVERY: AUTOMATION CLOUDAUTOMATION SUITESTANDALONERELEASE: 2023.4

## TABLE OF CONTENTS

### [Queues](#)

[General View](#)[Filters](#)[Queues Overview](#)[Unprocessed Items](#)[Transactions Overview](#)[Transactions Timeline](#)[Queues Details](#)[Individual View](#)[Filters](#)[Transactions Overview](#)[Unprocessed Items](#)[Queue Dynamics](#)[Retry Rate](#)[Transactions Timeline](#)[Processing Time](#)[Error Feed](#)

## Queues

### General View

Displays information about all the existing queues in the current folder on an aggregate basis, and allows you to check the overall health of the queues in your system. On this page, you can control the granularity of the displayed data, to have a more accurate overview of the

metrics of your queues.

## Filters

- **Interval** - enables you to control the granularity of the data, and display either the last hour or hour, last day (last 24 hours) hours), last week (last 7 days) or last month (last 30 days) of your queues activity.
- **Include Subfolders** - enables you to select if the contents of all subfolders are included. Only available for modern folders.

 **NOTE:**

Filtering impacts all widgets, unless specified otherwise in the dedicated section.

## Queues Overview

Allows you to see an overview of the queue's health state so you can assess if there are any problems that would impact the entire system. Each color block represents a specific queue from your instance. Hovering over a block displays the name of the corresponding queue and its containing folder. Note that it is possible to have multiple queues with the same name if each is in a different folder. The total number of queues is displayed in parentheses after the widget name. As a queue's state changes, the chart gets updated and the block color changes accordingly.

The various scenarios are represented with colored blocks and tooltips as follows:

Color	Tooltip
Grey	No changes detected
Green	Processing with no issues
Orange	Queue items soon overdue
Red	Queue items processing failed with application exceptions Queue items overdue

Clicking a block displays the corresponding detailed view page. Click the arrow at the top-left of the page to return to the general view.

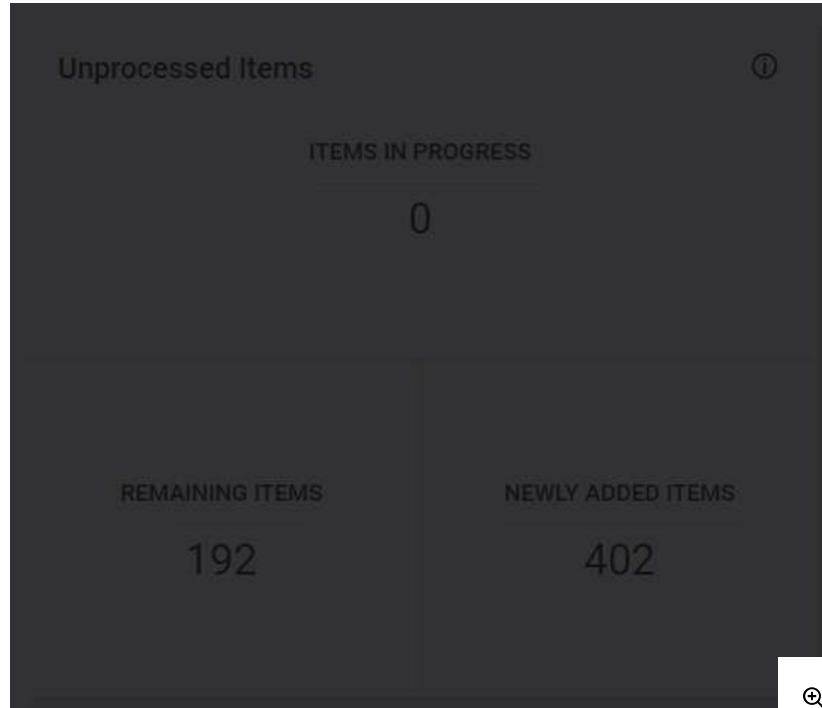
Filtering by **Interval** does not impact the red blocks in this widget.

## Unprocessed Items

Allows you to see the overall status of all queue items so as to assess if your system is overloaded, and more Robots are needed to process everything in time. Starting an item-processing job causes the **Items in Progress** section to update accordingly.

The unprocessed queue items are broken down into three categories:

- **Newly Added Items** - Items that were added in the selected interval regardless of their status.
- **Remaining Items** - Items remaining to be processed.
- **Items in Progress** - Items which are in progress.



Filtering by **Interval** only impacts the **Newly Added Items** section in this widget.

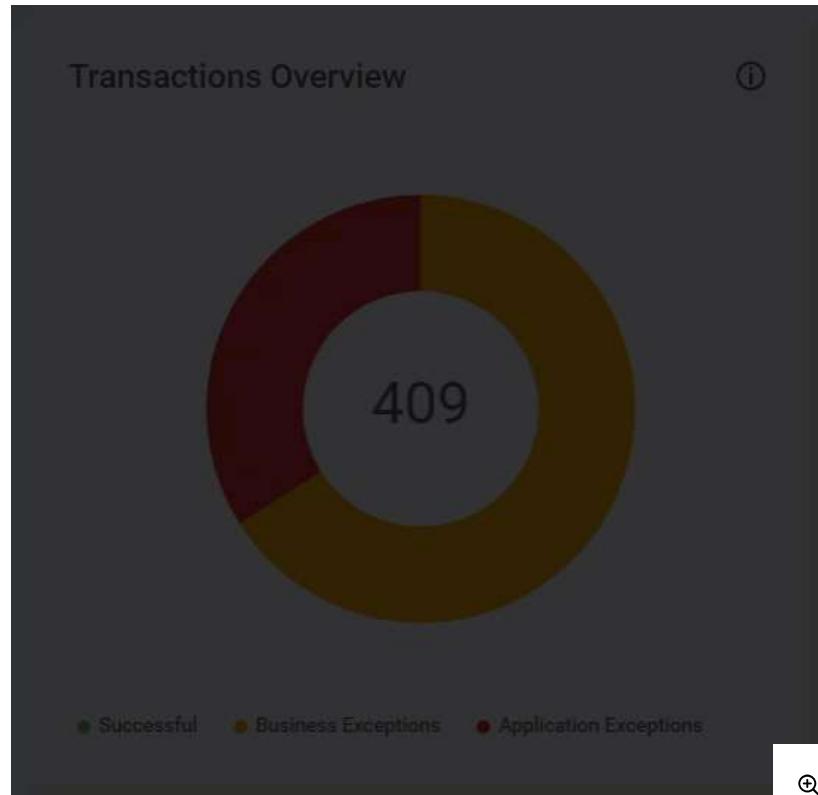
## Transactions Overview

Allows you to see precise figures on items that were processed. It enables you to assess whether your queue items are failing or not. Please note that for failed items that were retried, each retry status is taken into account.

**Transactions states** are represented with colors as follows:

Color	Tooltip
Green	Successful
Orange	Business Exceptions
Red	Application Exceptions

The number in the middle of the chart represents the total number of transactions as illustrated by this widget. The number on each slice represents the transactions in that specific state. Once an item is processed, successfully or not, the chart gets updated accordingly.



## Transactions Timeline

Allows you to see the timeline of processed items broken down according to their final status, in the selected interval. Please note that for failed items that were retried, each retry status is taken into account.

[Transaction states](#) are represented with colors as follows:

Color	Tooltip
Green	Successful
Yellow	Biz. Exception
Orange	App Exception
Red	Abandoned

Once an item-processing job has finished or it has been stopped, the chart gets updated accordingly. Click a marker in the chart to see the exact corresponding value.



## Queues Details

Allows you to see information pertaining to each queue and the corresponding items metrics. The total number of queues is displayed in parentheses after the widget name.

Field	Description
<b>Queue</b>	The name of the queue. Click the arrows at the right of the column name: <ul style="list-style-type: none"> <li>once to sort data by name, in ascending alphabetical order;</li> <li>twice to sort data by name, in descending alphabetical order.</li> </ul>
<b>No. of Items</b>	The total number in the queue. It is the same as on the <b>Queues</b> page.
<b>Deferred</b>	The number of queue items that have a <b>Postpone</b> date.
<b>Overdue</b>	The number of items that have past their established <b>Deadline</b> .
<b>On Time</b>	The number of items whose processing falls in the expected estimated time, between the <b>Postpone</b> (if any) and <b>Deadline</b> dates.
<b>AHT (Per Item)</b>	The average handling time of a processed item, calculated on all items in the selected queue since it was created. The time is displayed as a rounded-up value. The exact duration in ISO8601 format is shown when you hover over the displayed value.
<b>Completion</b>	The estimated completion time for the rest of the items based on the average handling time.
<b>Folder</b>	The folder where the Queue is located.

Field	Description
Search	Enables you to search by queue name.

Queues Details (3)							Search
QUEUE ^	NO. OF ITEMS	DEFERRED	OVERDUE	ON TIME	AHT (PER ITEM)	COMPLETION	
BestestQueue	0	0	0	0	N/A	N/A	
ReportsQueue	8	0	0	8	N/A	N/A	
TestingQueue	0	0	0	0	N/A	N/A	

Page 1/1

Filtering by **Interval** does not impact this widget.

## Individual View

Displays information concerning one queue, as selected by the user from the general view, specifically on the **Queues Overview** widget. Each widget in this view contains the exact same information as on the **Queues** page. Changing the properties there automatically causes the information here to update accordingly.

## Filters

**Interval** - allows you to control the granularity of the data, and display either the last hour or last day (last 24 hours) of the selected queue's health state.

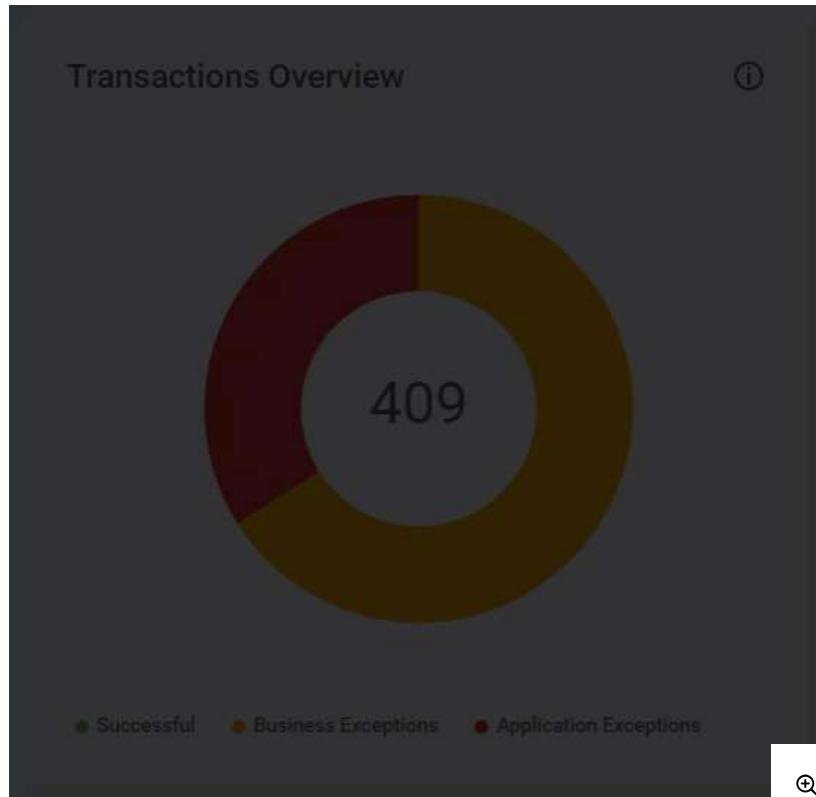
## Transactions Overview

Allows you to see precise figures on items that were processed in the selected queue. It provides metrics about the encountered exceptions so that you can assess whether your queue items are failing or not. Please note that for failed items that were retried, each retry status is taken into account.

**Transaction states** are represented with colors as follows:

Color	Tooltip
Green	Successful
Orange	Business Exceptions
Red	Application Exceptions

The number in the middle of the chart represents the total number of transactions as illustrated by this widget. Once an item is processed, successfully or not, the chart gets updated accordingly. You can filter out states by clicking the labels below the chart.

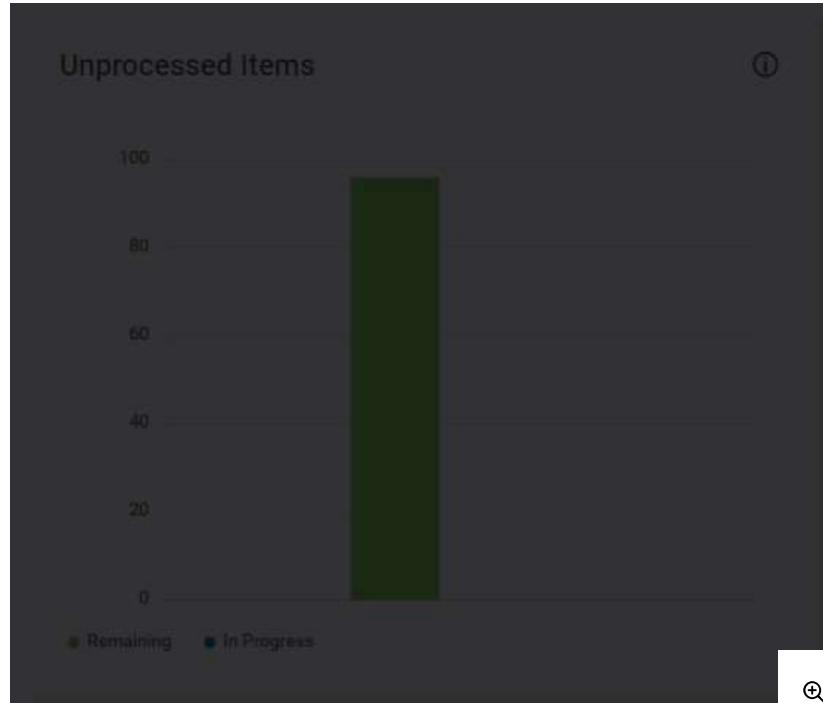


## Unprocessed Items

Allows you to see unprocessed items in a queue broken down into two categories:

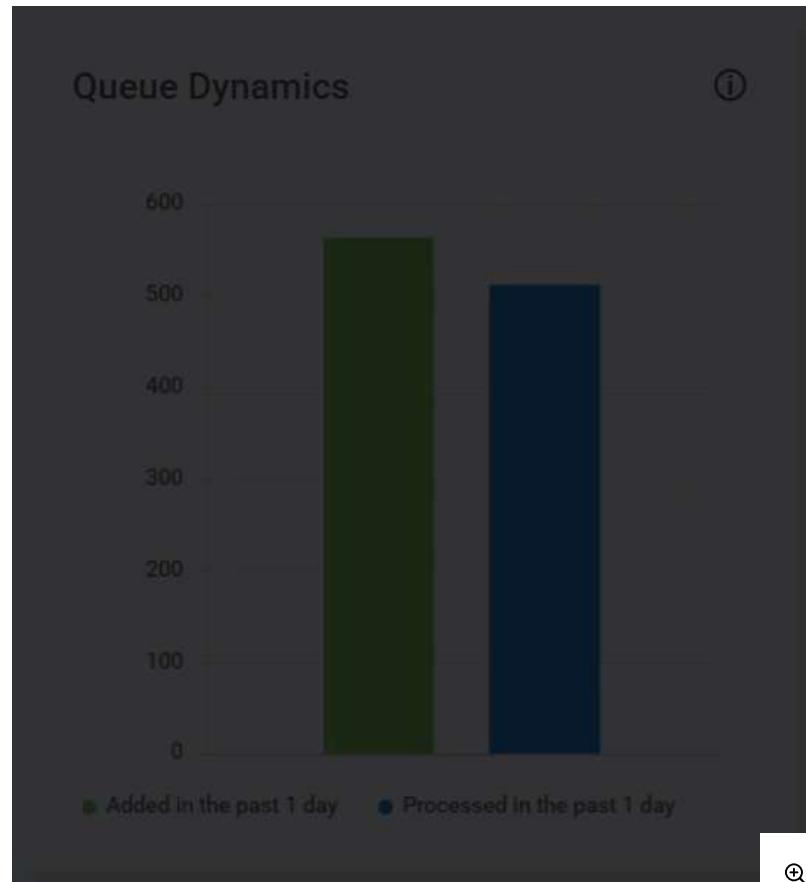
- Items remaining to be processed
- Items which are in progress.

This enables you to see the overall status of all the items in a queue so as to assess if your system is overloaded, and more Robots are needed to process everything in time. Starting an item-processing job causes the **Items in Progress** section to update accordingly. You can filter out states by clicking the labels below the chart.



## Queue Dynamics

Allows you to see the number of items in the selected queue that were either added or processed, in the selected interval, thus enabling you to assess the efficiency of your system concerning item processing. You can filter out states by clicking the labels below the chart.



## Retry Rate

Enables you to see your successfully processed queue items broken down according to the number of retries. The number of retries for successfully executed items are also taken into account in this widget. You can filter out states by clicking the labels below the chart.



## Transactions Timeline

Allows you to see the timeline of processed items in the selected queue, broken down according to their final status, in the selected interval. Please note that for failed items that were retried, each retry status is taken into account.

[Transaction states](#) are represented with colors as follows:

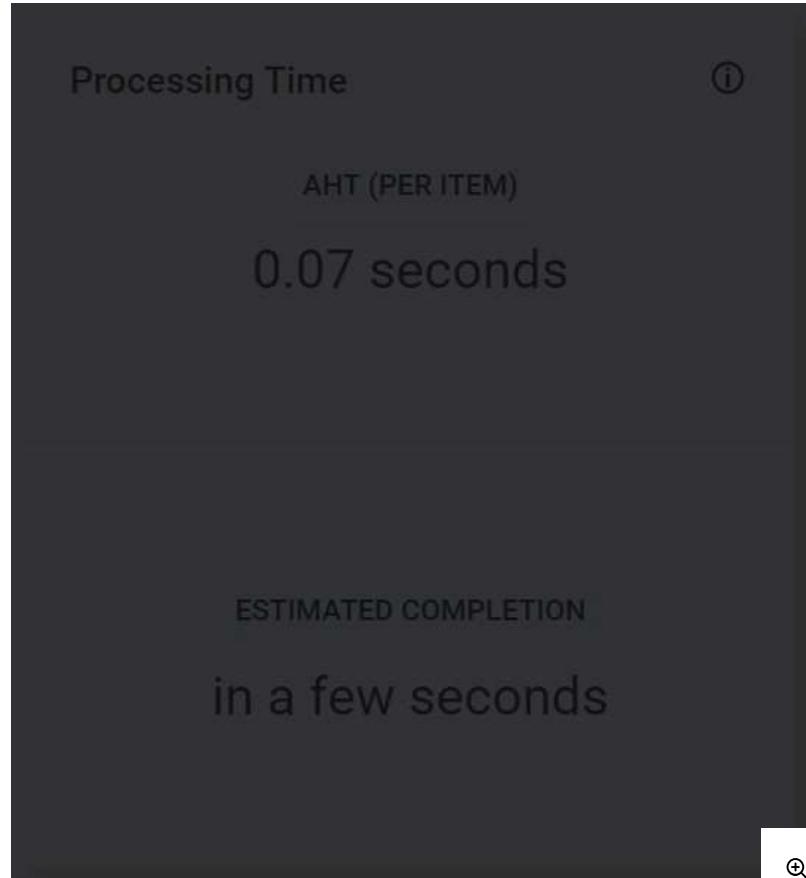
Color	Tooltip
Green	Successful
Yellow	Biz. Exception
Orange	App Exception
Red	Abandoned

Once an item-processing job has finished or has been stopped, the chart gets updated accordingly. Click a marker in the chart to see the exact corresponding value.



## Processing Time

Enables you to see the average handling time of an item, calculated on all items in the selected queue since it was created, and the estimated completion time for the rest of the items based on that average. The **Estimated Completion** value is influenced by the number of Robots which are currently processing items, as well as by items that have a **Defer Date**. When no more unprocessed items are left in the queue, the value is set to **N/A**.



Filtering by **Interval** does not impact this widget.

Note that the durations are displayed as rounded-up values. The exact duration in ISO8601 format is shown when you hover over the displayed value.

## Error Feed

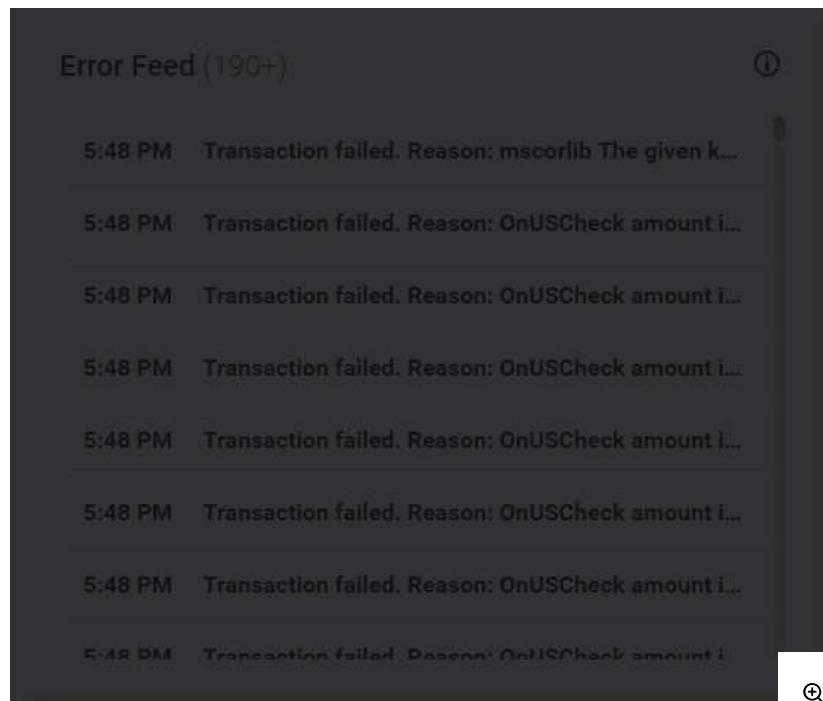
Allows you to see the a list of the errors encountered on the selected queue. The errors are displayed in reverse chronological order as they occur, meaning a new error is always displayed at the top of the feed. The number of errors displayed in parentheses after the widget name is expressed according to your position within the feed and to the total number of errors. Let's say there are 453 errors in your feed. The following behavior is encountered:

Position Within the Feed	Displayed Value
Between 0 and 99	<b>99+</b>
Between 100 and 199	<b>199+</b>
Between 200 and 299	<b>299+</b>
Between 300 and 399	<b>399+</b>
Further than 400	The total number of errors in the feed, in this particular example <b>453</b> .

After reaching the end of the feed, the total number of errors is displayed regardless of your position.

You can dismiss an error after having acknowledged it, however please take into account that multiple users might check the feed simultaneously, so we recommend carefully using the **Dismiss** button. A dismissed error is still taken into account in the error counter. After an error was dismissed, the name of the user who performed the action is displayed adjacently in a tooltip. Note that you need **Edit** permissions on Monitoring to dismiss errors.

Field	Description
Time	The time when the error occurred.
Error Message	The actual error message.
Dismiss	Enables you to dismiss the specific error message. Clicking the <b>Dismiss</b> button removes the formatting from the text. A dismissed error is still taken into account in the error counter. After an error was dismissed, the name of the user who performed the action is displayed adjacently. Note that you need <b>Edit</b> Permissions on Monitoring in order to dismiss errors.
Copy	Enables you to copy the text of the error message to the clipboard.



Default Theme

English

# Orchestrator Standalone User Guide

DELIVERY: AUTOMATION CLOUDAUTOMATION SUITESTANDALONERELEASE: 2023.4

## TABLE OF CONTENTS

### [Queue Item Statuses](#)

[Item Statuses](#)[Revision Statuses](#)[Statuses Diagram](#)

# Queue Item Statuses

Queue items can have two types of statuses:

1. Item Statuses
2. Revision Statuses

## Item Statuses

These statuses let you know if an item has been processed or not, and the stage of the process at a particular time. Item statuses are displayed in the **Status** column, in the **Transactions** page. Queue items can go through the following statuses:

- **New** – the item has just been added to the queue with the **Add Queue Item** activity, or the item was postponed, or a deadline was added to it, or the item was added after an attempt and failure of a previous queue item with auto-retry enabled.
- **In Progress** – the item was processed with the **Get Transaction Item** or the **Add Transaction Item** activity; when an item has this status, your custom progress status is also displayed, in the **Progress** column;
- **Failed** – the item did not meet a business or application requirement within the project and was therefore sent to a **Set Transaction Status** activity, which changed its status to Failed;
- **Successful** – the item was processed and sent to a **Set Transaction Status** activity, which changed its status to Successful;
- **Abandoned** – the item remained in the **In Progress** status for a long period of time (approx. 24 hours) without being processed;
- **Retried** – the item failed with an application exception and was retried. After the Robot finishes retrying the item, the status changes to Failed or Successful, according to your workflow.
- **Deleted** – the item has been manually selected from the **Transactions** page and marked as deleted; an item with this status can no longer be processed.

**⚠️ IMPORTANT:**

To support our effort of consolidating queue item final statuses, you can no longer use the `SetTransactionResult` endpoint to:

- change a transaction payload once it reaches a final state (be it **Failed** or **Successful**)
- rerun a transaction by using the `DeferDate` and `DueDate` properties, in order to move it out of a final state (be it **Failed**, **Successful**, **Abandoned**, or **Deleted**)

## Revision Statuses

These statuses let you perform version control but **only of queue items that have been abandoned or have failed with an application or business exception**. These statuses have to be manually set per item, by an assigned reviewer. All changes are tracked in the **History** tab of the **Audit Details** window. The reviewer can be assigned only when the item status is failed or abandoned, and reviewers cannot be changed after a revision status was added to the item. Only logged in reviewers can see requests assigned to them in the **Review Requests** page. Moreover, queue items can be assigned for revision in bulk.

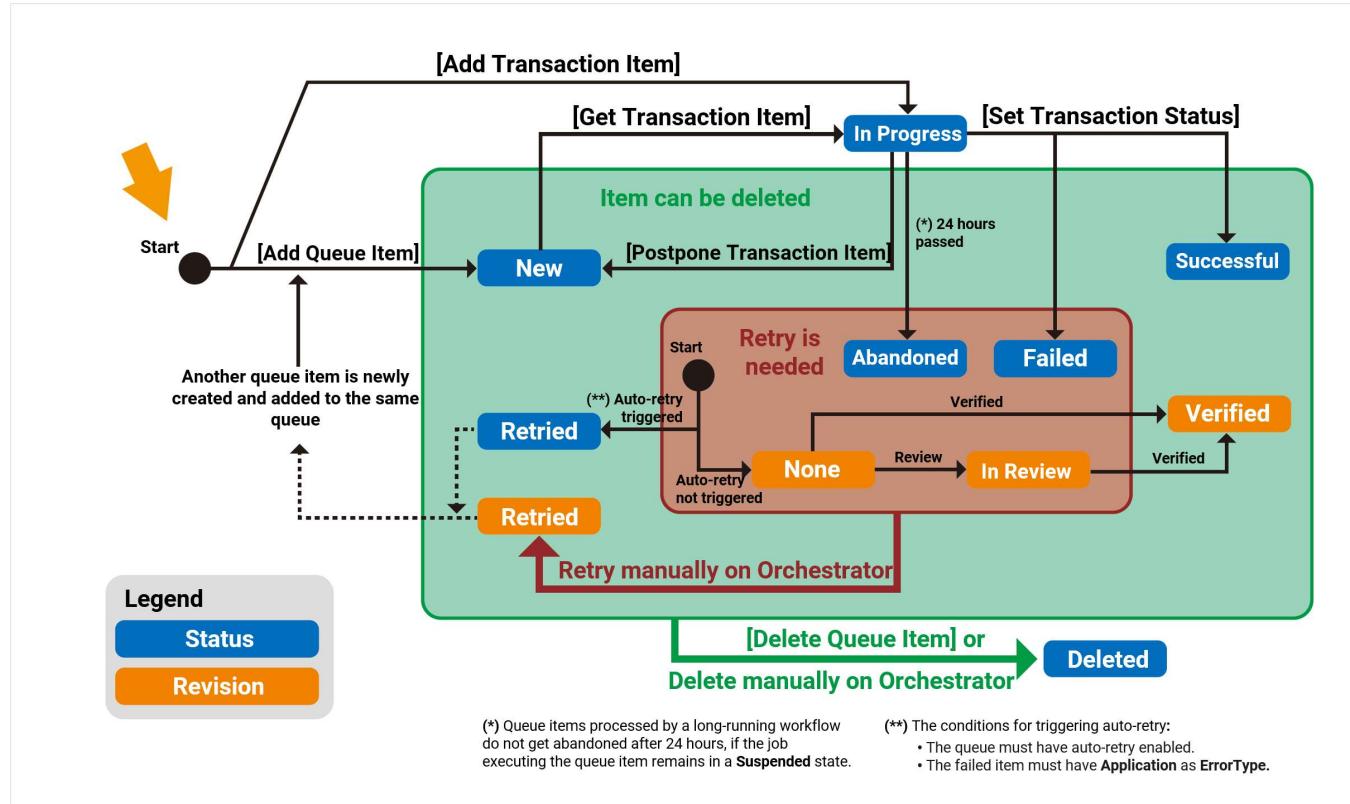
DETAILS		COMMENTS		HISTORY		
ACTION	WHO	STATUS	REVIEW	REVIEWER	TIME	C
Edit	admin	Failed	None	Documentation	29 minutes ago	
Edit	Documentation	Failed	None	vlad.tudoran	2 minutes ago	
Edit	Documentation	Failed	None		a few seconds ago	
Items: 10		◀ < Page 1/1 > ▶				3 items
						CLOSE

The following statuses are available:

- None** - this is the default status. It is set to all items, even if they failed or not.
- In Review** - a user has marked an item that has failed with app exception as in the process of being reviewed. This status does not have other implications in Orchestrator or Studio than changing the value in the **Revision** column on the **Queues** page.
- Verified** - a user has marked an item as verified. Items cannot be retried after the user sets this status. There are no other implications in Orchestrator or Studio than changing the value in the **Revision** column on the **Queues** page.
- Retried** - the item has been marked manually for retry. As a result, a new queue item with the **New** status is created. This is displayed in the **Items Details** window of the indicated transaction.

## Statuses Diagram

The diagram below explains the queue items transition through each status.



Default Theme

English

# Orchestrator Standalone User Guide

DELIVERY:

AUTOMATION CLOUD

AUTOMATION SUITE

STANDALONE

RELEASE:

2023.4



## TABLE OF CONTENTS

### [Job States](#)

# Job States

Jobs can have the following states:

-  **Pending** - A job is in this state if it is queued on the same Robot or it is trying to establish a connection with the Robot (only different jobs on the same Robot can be queued).
-  **Running** - A job is in this state if it has established a connection to the Robot which started executing the designated process.
-  **Successful** - A job is in this state if it has been executed correctly by the Robot and it has finished running or has been stopped using the **Stop** button.
-  **Faulted** - A job is in this state if it failed to start or the process threw an unhandled error during execution.
-  **Stopping** - An intermediary state that is triggered if you click the **Stop** button in Orchestrator. The job is canceled as soon as it is safe. This can be implemented in a workflow using the [Should](#)

Stop activity.

-  **Terminating** - An intermediary state that is triggered if you click the **Kill** button in Orchestrator. By default, a cleanup background job runs once every three hours and transitions to Failed the jobs that have been in a Terminating state for at least one day.
-  **Suspended** - An intermediary state that is triggered with the purpose of allowing user intervention or completion of an intermediary process. It is triggered by the corresponding activities in Studio. Details [here](#).
- **Resumed** - An intermediary state that is triggered if the conditions (user intervention, intermediary process completion) of a fragmented workflow have been met. It is triggered by the corresponding activities in Studio. Details [here](#).
-  **Stopped** - A job is in this state if it stopped (by using the **Kill** button, or by canceling it from the system tray) before it finished executing without throwing any errors.

To view more information about all job executions, click the corresponding **Details** button. The **Job Details** window is displayed and enables you to view why a job faulted. Additionally, for unattended faulted jobs, if your process had the **Enable Recording** option switched on, you can also [download](#) the corresponding execution media and check the last moments of the execution before failure.

## Job Details

**Process:** TestingSequence  
**Environment:** DOC  
**Robot:** Gustave  
**Machine name:** CAPETRINA  
**Info:** Job completed  
**Start Time:** 05/09/2019 7:36:57 PM  
**End Time:** 05/09/2019 7:37:02 PM  
**Input Values:** null  
**Output Values:** Empty  
**Has Recording:** false

CLOSE





Default Theme

English



# Orchestrator Standalone User Guide

DELIVERY:

AUTOMATION CLOUD

AUTOMATION SUITE

STANDALONE

RELEASE:

2023.4



## TABLE OF CONTENTS

### [About Assets](#)

[Asset Types](#)[Asset Values](#)[Global Value](#)[Value Per Account-Machine](#)[Value Per Account](#)

# About Assets

Assets usually represent shared variables or credentials that can be used in different automation projects. They allow you to store specific information so that the Robots can easily access it.

Additionally, an extra level of security is provided, as all assets of type Credential stored here are encrypted with the AES 256 algorithm. They can be invoked by RPA developers when designing a

process, but their values can be hidden from them.

The **Assets** page enables you to create new assets. It also displays all previously created assets, which can be edited or deleted.

The **Get Asset** and **Get Credential** activities used in Studio request information from Orchestrator about a specific asset, according to a provided **AssetName**. If the **AssetName** provided in Studio coincides with the name of an asset stored in the Orchestrator database, and the Robot has the required permissions, the asset information is retrieved and used by the Robot when executing the automation project.

## Asset Types

There are four types of assets:

- **Text** - stores only strings (it is not required to add quotation marks)
- **Bool** - supports true or false values
- **Integer** - stores only whole numbers
- **Credential** - contains usernames and passwords that the Robot requires to execute particular processes, such as login details for SAP or SalesForce.

 **NOTE:**

All asset types are encrypted in the Orchestrator database by default.

Existing assets are also encrypted when updated.

## Asset Values

The value assigned to an asset can have either or both of the following value types:

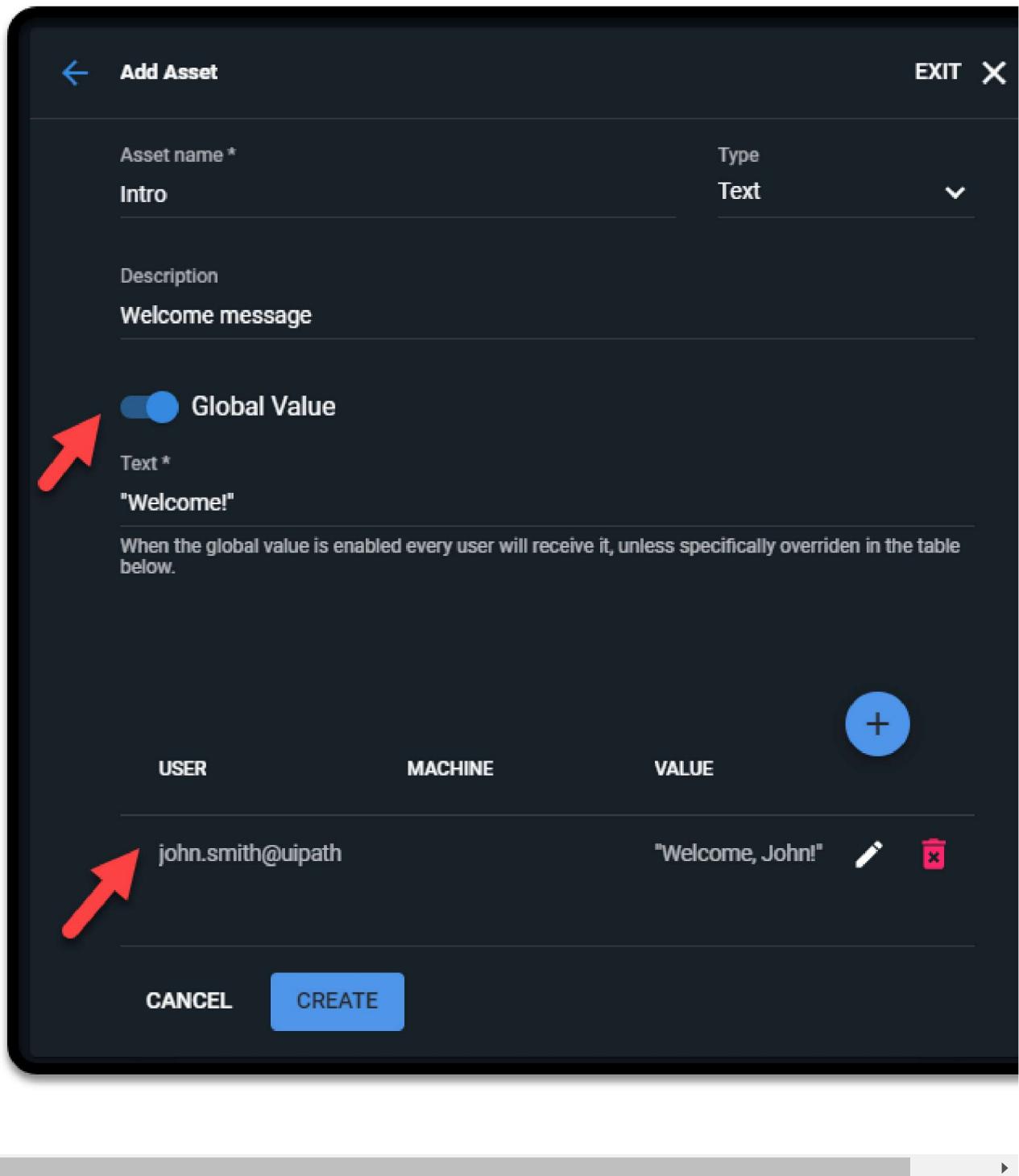
### Global Value

All accounts receive the asset value unless a specific value is assigned on a per-account basis.

- All accounts receive the asset value.

The screenshot shows the 'Add Asset' screen in the UiPath Orchestrator application. At the top, there is a back arrow labeled 'Add Asset', a 'EXIT' button with a close icon, and a dropdown menu. Below this, the asset name is set to 'Intro' and the type is 'Text'. A red arrow points to the 'Global Value' toggle switch, which is turned on. The 'Description' field contains 'Welcome message'. The 'Text' field contains the value '"Welcome to UiPath!"'. A note below states: 'When the global value is enabled every user will receive it, unless specifically overridden in the table below.' At the bottom, there is a table header with columns 'USER', 'MACHINE', and 'VALUE', and a blue '+' button. A note says: ':≡ Optionally you can also add specific values per user.' At the very bottom are 'CANCEL' and 'CREATE' buttons.

- All accounts receive the global value, except for John Smith, who receives a particular value only defined for their account.



## Value Per Account-Machine

The value is received only by the specified account-machine pair. When specifying account-machine pairs, make sure they are valid i.e., the configured [account-machine mappings in the tenant or folder](#) do not exclude them, otherwise job execution is not possible on that pair, hence the asset does nothing.

**⚠️ IMPORTANT:**

- Assets per user-machine only work on Robots v20.10 or newer.

Add Asset

Asset name \*

SAPCredential\_JS\_fin

Type

Credential

Description

Credential finance.

Credential Store

Orchestrator Database

Global Value

Username

Password

When the global value is enabled every user will receive it, unless specifically overridden in the table below.

USER	MACHINE	VALUE	EDIT	DELETE
john.smith@uipath	Finance	john.smith:****		

**CANCEL** **CREATE**

## Value Per Account

The value is received only by the specified account. If **Global Value** is disabled, at least one per-account value must be provided.

Asset name \*

SAP Credential

Type

Credential

Description

Credential John Smith

Credential Store

Orchestrator Database

Global Value

Username

Password

When the global value is enabled every user will receive it, unless specifically overridden in the table below.

USER	MACHINE	VALUE
petrina.calota@uipath		john.smith:****

**NOTE:**

**NOTE:**

Upon upgrading to v2020.10, existing assets in modern folders are migrated to assets using a

global value and no per-account value.



---

Default Theme

English

# Orchestrator Standalone User Guide

DELIVERY: AUTOMATION CLOUD AUTOMATION SUITE STANDALONERELEASE: 2023.4 ▾

## TABLE OF CONTENTS

### Managing Assets in Orchestrator

[Creating Assets](#)[Managing Asset Links](#)[Linking Multiple Assets to the Current Folder](#)[Linking an Asset to Multiple Folders](#)[Unlinking Assets From Folders](#)[Editing Assets](#)[Adding Tags to Assets](#)[Removing Tags From Assets](#)[Removing Assets](#)

# Managing Assets in Orchestrator

## Creating Assets

1. In the **Assets** page, click **Add**. The **Add Asset** window is displayed.
2. In the **Asset name** field, enter a name for the asset.
3. From the **Type** list, select the type of asset you want to create.
4. Add a **Description** of the asset.
5. From the **Credential Store** list, select where this asset is stored from any of your defined credential stores.
6. Use the **Global Value** toggle to indicate whether this asset has a default value (used when no specific per-account/per-account-machine value exists).
  - If enabled, in the **Value** field, type what the asset should contain. If the asset type is boolean, select **True** or **False**, and if it is a credential enter the **Username** and **Password**.
  - If disabled, you must provide at least one account-specific asset value.
7. Use the **Add robot asset value** button to create the per-account or per-account-machine asset values desired.

8. Click **Create**. The asset is created and displayed on the **Assets** page.

**ⓘ NOTE:**

Please note that once an asset is created, the asset type cannot be changed.

Field	Description
<b>Asset name</b>	A custom name for the asset to help you identify it easily. Cannot exceed 256 characters.
<b>Type</b>	The type of the asset. The following options are available: <ul style="list-style-type: none"> <li><b>Text</b> - assets that can store string values. This is the default option.</li> <li><b>Bool</b> - assets that can store a true or false value. If you select this option, the <b>Value</b> field displays two radio buttons (<b>True</b>,<b>False</b>).</li> <li><b>Integer</b> - assets that can store integers.</li> <li><b>Credential</b> - assets that can store credentials. If you select this option the <b>Value</b> field is replaced by two others: <b>Username</b> and <b>Password</b>.</li> </ul>
<b>Credential Store</b>	Available only when adding credential assets. The credential store where the asset will be saved, whether the native Orchestrator database or any of your created external stores.
<b>External Name</b>	Available only when adding credential assets to a credential store. The name of the asset to be used when communicating with the credential store.
<b>Global Value</b>	An asset with a global value asset is received by all accounts, except those with specific asset values (defined using the <b>Add robot asset value</b> button). When this option is enabled, you must fill in the global value that is received by accounts: <ul style="list-style-type: none"> <li>For <b>Bool</b> assets, two radio buttons, <b>True</b> and <b>False</b>, are displayed. By default, <b>False</b> is selected.</li> <li>For <b>Credential</b> assets the <b>Username</b> and <b>Password</b> fields are enabled. Both these fields have to be filled in.</li> <li>For text assets, this field allows for up to 1.000.000 characters.</li> </ul>
<b>Add robot asset value</b> 	Enables you to add specific asset values on a <a href="#">per account</a> or <a href="#">per account-machine</a> basis. When not using the <b>Global Value</b> option, you must configure at least one specific value. If a global value is provided, adding specific values is optional. <ul style="list-style-type: none"> <li>For <b>Bool</b> assets, two radio buttons, <b>True</b> and <b>False</b>, are displayed. By default, <b>False</b> is selected.</li> <li>For <b>Credential</b> assets the <b>Username</b> and <b>Password</b> fields are enabled. Both these fields have to be filled in.</li> <li>For text assets, this field allows for up to 1.000.000 characters.</li> </ul>

## Managing Asset Links

Sharing assets between folders enables launching jobs in multiple folders without redesigning your workflows in Studio when the underlying processes are targeting the same asset. Linking an asset to a folder makes the asset and all asset-associated objects, such as asset items, available in that folder.

**⚠️ IMPORTANT:**

You can only share assets with global values and no specific value per account. You cannot share assets with values per account between folders.

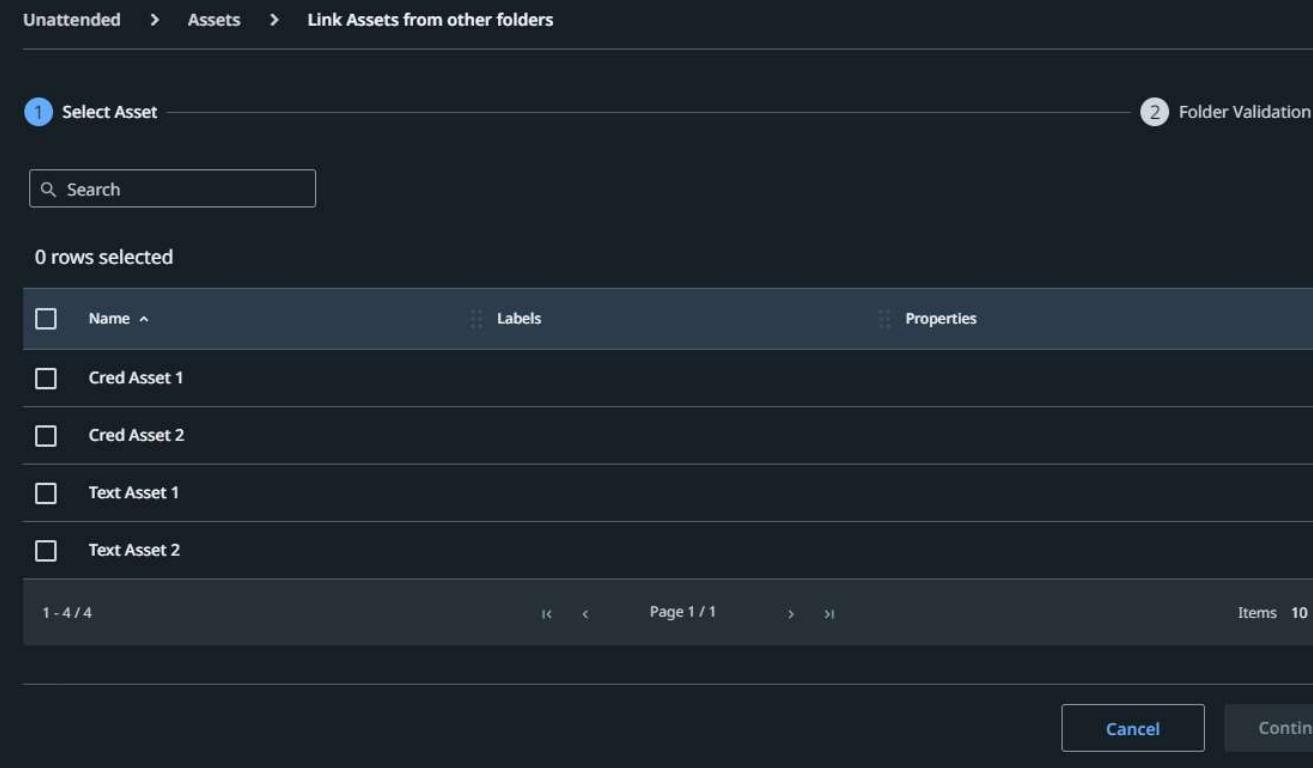
**ⓘ NOTE:**

An asset linked to multiple folders is marked using the  icon. If the icon is not present, then the current folder is the only folder the asset resides in. Deleting it here completely removes the asset from Orchestrator.

You need the **Assets - Create** permission in the folders where you want to add the asset (target folders) and **Assets - View** in the folder where the asset currently resides (original folder). If you have **Assets - Edit** in the target folder, you also require **Assets - Edit** in the original folder.

## Linking Multiple Assets to the Current Folder

1. In the folder you want to link an asset to, on the **Assets** page, click **Add**. Three buttons are displayed allowing you to add an asset, link assets from other folders, or hide the options.
2. Click **Link from other folders**. The **Link Assets** window is displayed showing a list of all assets in the folders in which you have View permissions on Assets.



The screenshot shows the 'Link Assets from other folders' window. At the top, there's a breadcrumb navigation: 'Unattended > Assets > Link Assets from other folders'. Below this, a section titled '1 Select Asset' contains a search bar with the placeholder 'Search' and a note '0 rows selected'. A table lists five assets: 'Cred Asset 1', 'Cred Asset 2', 'Text Asset 1', and 'Text Asset 2'. Each asset has a checkbox next to its name. To the right of the table are columns for 'Labels' and 'Properties'. At the bottom of the table area, there are pagination controls showing '1 - 4 / 4' and 'Page 1 / 1'. On the far right, there are 'Cancel' and 'Continue' buttons. A horizontal scrollbar is visible at the bottom of the window.

3. On the **Select Assets** section, select one or multiple assets from the list.
4. Click **Continue**. You are directed to the **Folder Validation** section. Here you can see the folders the assets are already linked to. If there are multiple folders, their names are displayed.

The screenshot shows a dark-themed dialog box titled "Link Assets from other folders". At the top left is a "Select Asset" button with a pencil icon. At the top right is a "Folder Validation" button with a blue circle containing a question mark. The main area contains a table with two rows:

Name	Folders
Cred Asset 1	@uipath.com's workspace, Important
Text Asset 1	Shared, @uipath.com's workspace, Important

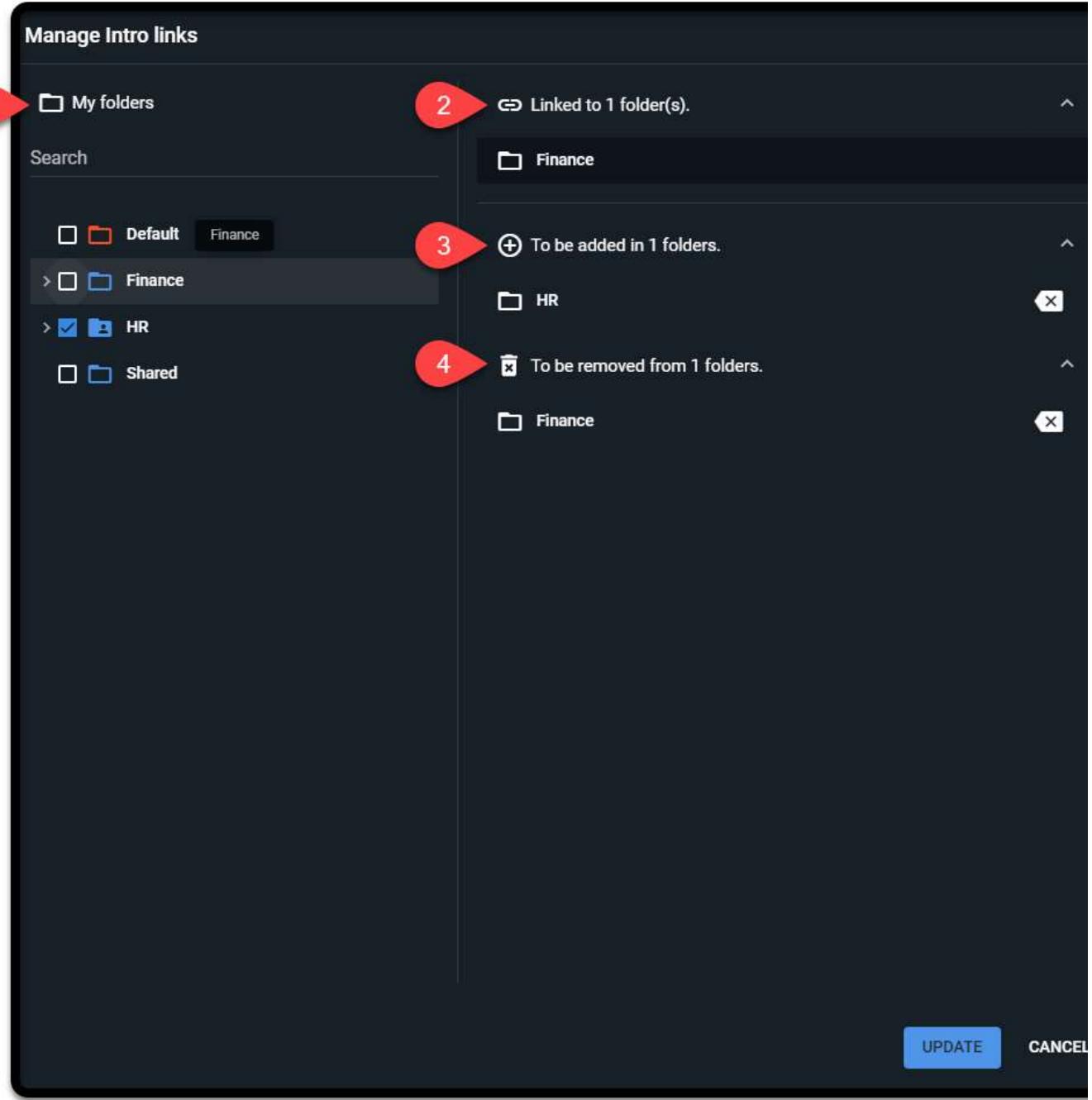
At the bottom right are "Cancel" and "Link" buttons. A horizontal scrollbar is visible at the bottom of the dialog.

5. Click **Remove** for the corresponding asset to revert the change or click **Exit** to cancel the operation.
6. Click **Link** if you want to make the link between the assets you selected and the current folder. The assets are displayed on the **Assets** page.

## Linking an Asset to Multiple Folders

1. Navigate to a folder the asset to be linked resides in.

Click **More Actions > Manage Links** for the desired asset to open the **Manage Links** window. The **Manage Asset Links** window is displayed.



1 - Left-hand pane displaying all the folders you have been granted **View** permissions on Assets.

2 - The current state of the asset displaying the number of folders it currently resides in as well as their names.

3 - The folders the asset is to be added in according to your selection in the left-hand pane.

4 - The folders the asset will be removed from.

1. Click **Update**. A confirmation window is displayed.

2. Click **Cancel** if you want to abort the changes or **Continue** for the changes to take effect. The operations are now reflected in Orchestrator according to your changes.

## Unlinking Assets From Folders

Unlinking assets from folders can be performed in a manner similar to the linking operation. Navigate to the link-management areas presented in the procedures above and remove the connections between a certain asset and a certain folder.

Alternatively, you can remove an asset using the **Remove** functionality. See [Removing an Asset](#).

**⚠️ IMPORTANT:**

Removing an asset that exists in multiple folders only removes it from the folder where the removal operation takes place, it does not remove it from the other folders as well. In order to completely delete an asset, you must remove all its existing links.

## Editing Assets

To edit an asset, click **More Actions > Edit** button, make the necessary changes and click **Update** in the **Update Asset** window. Only the asset value and name can be edited. All changes are audited and can be viewed in the [Audit](#) page. If you change the username of a credential asset, you have to also change the password.

## Adding Tags to Assets

**ⓘ NOTE:**

You need **Edit** on Assets and **View** on Tags to add existing tags to assets.

You need **Edit** on Assets and **Create** on Tags to add new tags to assets.

**ⓘ NOTE:**

- Each asset can have a maximum of one million key/value pairs.
- Labels and key/value properties are limited to 256 characters.
- Tag names can't contain these characters: <, >, %, &, ?, /, ::

You can apply tags to a asset either when creating one or editing an existing one. To add tags to a asset when editing it, follow these steps:

1. From the **Assets** page, click **More Actions > Edit** next to the desired asset. The asset is opened for editing.
2. On the **Labels** field, start typing the name of the label. You can choose an existing label or create a new one.
3. On the **Properties (key-value pairs)** field, click **Add new**.
4. Add new keys and values. You can choose existing keys and/or values or you can create new ones.
5. When done, click **Update**. Your asset is updated and the newly created tags, if any, become available for other objects.

## Removing Tags From Assets

To remove tags from a asset, follow these steps:

1. From the **Assets** page, click **More Actions > Edit** next to the desired asset. The asset is opened for editing.
2. On the **Labels** field, click the **X** adjacent to the name of the label to remove it. The label is removed.
3. On the **Properties (key-value pairs)** field, click the **X** adjacent to the keys and/or values to remove them. The keys and/or values are removed.
4. To delete a key/value pair click the **Remove** icon corresponding to that entry. The key/value pair is removed.
5. When done, click **Update**. Your asset is updated and tags are removed.

## Removing Assets

To remove an asset, click **More Actions > Remove**.

Alternatively, you can select one or multiple assets and click the **Remove** button.



Default Theme

English



# Orchestrator Standalone User Guide

DELIVERY:

AUTOMATION CLOUD

AUTOMATION SUITE

STANDALONE

RELEASE:

2023.4



## TABLE OF CONTENTS

### [Managing Assets in Studio](#)

[Using the Get Asset Activity](#)

[Using the Get Credential Activity](#)

# Managing Assets in Studio

Studio has two activities, **Get Asset** and **Get Credential**, that the Robot can use to extract information from assets stored in the Orchestrator database.

They are displayed in the **Activities** panel, under **Orchestrator > Assets**, and are part of the UiPath Core activities pack.

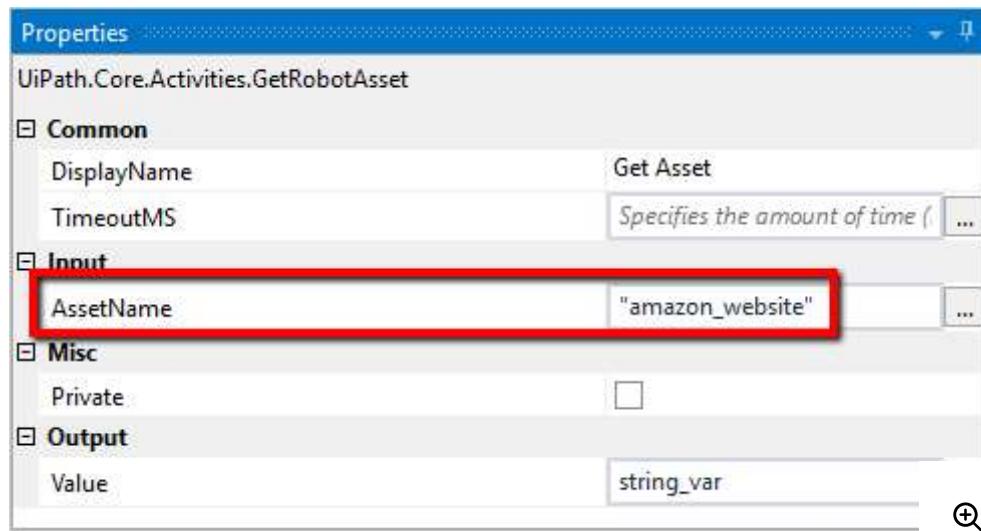
The **Get Asset** activity should be used with the String, Boolean and Integer assets, while **Get Credential** should be used with Credential assets.

**NOTE:**

The asset name is not case sensitive. For example, "Text" and "teXt" are the same.

## Using the Get Asset Activity

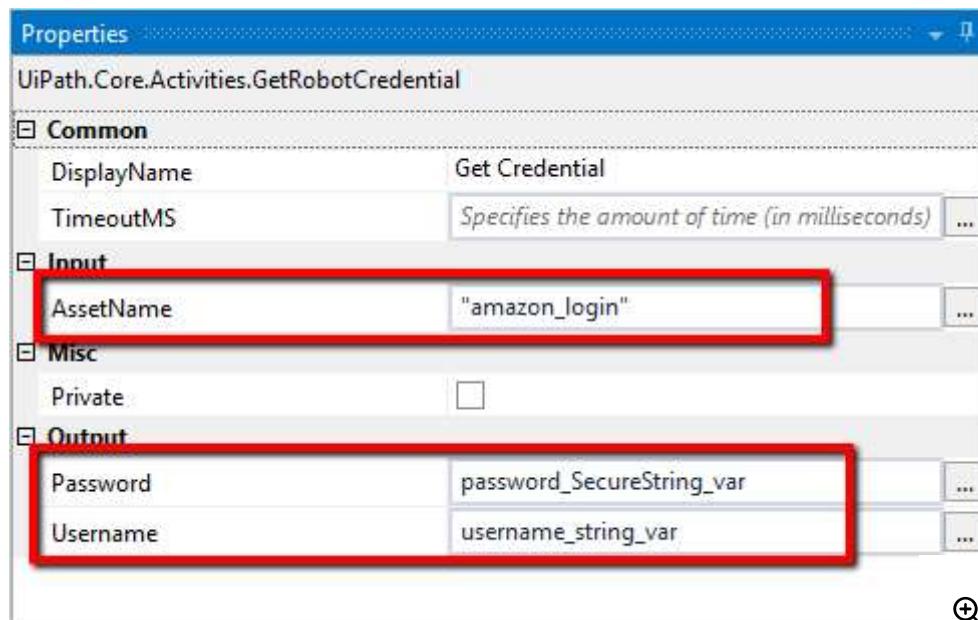
1. In Orchestrator, create a string, boolean or integer asset.
2. In Studio, create a string, boolean or integer variable, depending on the type of asset you want to use from Orchestrator. For example, if you want to work with a string asset, create a string variable.
3. From the **Activities** panel, drag a **Get Asset** activity to the **Main** panel.
4. In the **Properties** panel, in the **AssetName** field, type the name of the Orchestrator asset you want to use, and place it between quotation marks. For example, if in Orchestrator the asset name is `StringAsset`, write `"stringasset"` in Studio.
5. In the **Properties** panel, in the **Value** field, enter the variable created at step 2. This variable stores information from the specified Orchestrator asset, if the **AssetName** coincides with the asset name that is stored in the Orchestrator database, and the Robot has the required permissions.



## Using the Get Credential Activity

1. In Orchestrator, create a **Credential** asset.
2. In Studio, create a String variable. This is used to store the username part of the credentials.

3. Create a SecureString variable. This is used to store the password. The SecureString is a special .NET Framework variable type that is encrypted within the framework.
4. In the **Properties** panel, in the **AssetName** field, type the name of the credential asset, as it is in Orchestrator, and place it between quotation marks. For example, "amazon\_login".
5. In the **Properties** panel, in the **Password** field, enter the SecureString variable.
6. In the **Properties** panel, in the **Username** field, enter the string variable. The credential asset stored in Orchestrator can be used by the Robot too, as long as the **AssetName** coincides with the asset name that is stored in the Orchestrator database, and the Robot has the required permissions.



Default Theme

English



# Robot User Guide

RELEASE:

2023.4



## TABLE OF CONTENTS

### [Background Process Automation](#)

Attended and Unattended Robots Behavior

Attended Robots

Unattended Robots

Running Background Processes

# Background Process Automation

#### NOTE:

This is only available for **Attended Robots**. When using **Unattended Robots** to run multiple Processes at the same time, each Running Process consumes a separate license.

An Attended Robot operates on the same machine as a human. It is usually triggered by user events and should only be run under human supervision. Read more about the Robot's automation capabilities [according to license](#).

Background Process Execution is performed by a Robot on a machine for a particular user. Processes run in the same Windows session at the same time.

Each running process uses its own version of dependencies, even if multiple running processes require a different version of the same dependency. For example, if **process A** requires **dependency v18.4.6** and **process B** requires **dependency v19.4.4**, then each process downloads and uses its required version of the dependency.

According to this type of license, an Attended Robot can concurrently execute one foreground process (with UI interaction) and multiple background processes (without UI interaction).

A foreground process is used when your automation project needs to interact with UI elements. These processes heavily rely on UIAutomation activities. Please note that you can only execute one foreground process at a time.

#### **IMPORTANT:**

Automation processes that use UIAutomation activities cannot run under a locked screen.

A **background process** doesn't need to interact with UI elements on the screen but rather relies on background processes to pass along information. UIAutomation activities should not be used in these project types. Multiple such processes can simultaneously run on a machine, even if a foreground process is already running.

The type of process is generally dictated by the type of activities it uses (whether or not they interact with UI elements). However, you can specify the type when you create a new process from Studio. By default, all processes are marked as foreground, unless specified otherwise.

## Attended and Unattended Robots Behavior

During start-up, a Windows Session is created for the System Processes to run which are not related to a specific user, this is called **Session 0**. This session allows Windows to run system processes needed for the machine itself. When a user logs on to that machine, a new session is created called **User Session** in which user-specific services run.

For a more detailed description, the [Session 0 Isolation](#) document from Microsoft provides additional information on how sessions are created and information is handled between them.

Using background processes offers a different behavior when they are run in Attended vs Unattended automations. Background Processes started from an Attended Robot run in the user's session (**Session 1**) while the ones started from Unattended Robots run in the Windows Session.

## Attended Robots

Background Process Automation with Attended Robots rely on the user's session on the machine to run automations, these are started from Studio or UiPath Assistant and run in the same Windows Session as the user. As the automation happens in the same session, the robot is able to retrieve information and access files that are specific to the user.

## Unattended Robots

Background Process Automation with Unattended Robots are started from via the Robot Service and run in the Windows Session (Session 0). It's important to know that Session 0 has no User Interface and cannot interact with a user session. When using this type of processes on Unattended Robots keep in mind that they run in the name of the user and they inherit its permissions. Make sure that the user under which the process runs has access to the needed resources and that the process does not require any type of User Interface. See the [Windows Session](#) document for more details on how windows sessions are being handled by the UiPath Robots.

 **NOTE:**

When running Background Automations via Unattended Robots, using Microsoft Office applications such as Word, Excel or PowerPoint might not work properly as they have been built to run in User-Session, meaning that they need an interactive desktop session and user profile. For additional information on how Office 365 Products work with Server Sessions(session 0), see the [Server-side Automation of Office](#) document from Microsoft.

## Running Background Processes

A **Background Process** can be transitioned to **Foreground** with the help of **Use Foreground Activity**. More details on how this activity works and how to use it in building your automation can be found [here](#).

As long as at least one process is running, the Robot is marked as **Busy** in Orchestrator.

 **IMPORTANT:**

Starting with the 2021.10 release, running unattended background automations on service-mode robots run by default under the built-in "Local Service" Windows user. Prior to this, the username and password configured for the robot in Orchestrator were used.

To use credentials specified in Orchestrator, you need to configure the `UIPATH_HEADLESS_WITH_USER` system environment variable on the robot machine and set the value to True.

Running automations under the Local Service account is lightweight as it does not need login credentials set in Orchestrator. This has the following limitations:

- In automation workflows, access is limited to Local Service account privileges.
- Proxy configurations for the executor are not supported.

Starting another foreground process while one is already running is not possible. Background processes, on the other hand, can be started regardless of the state and type of the other processes.

You can use the [Run Parallel Process](#) activity in your background processes to start other processes, based on triggers you need. Processes started by this activity run independently. For example, a background process which monitors particular actions can start a new process (foreground or background) and continue to monitor the specific action without being interrupted. A background process loaded by this activity starts right away, even if a foreground and one or more background

processes are already running. If the activity starts a foreground process, one of the following situations occurs:

- The process starts right away if there is no other foreground process running.
- An exception is thrown if a foreground process is already running.



Default Theme

English

# Studio User Guide

RELEASE:

2023.4



## TABLE OF CONTENTS

### [Background Process](#)

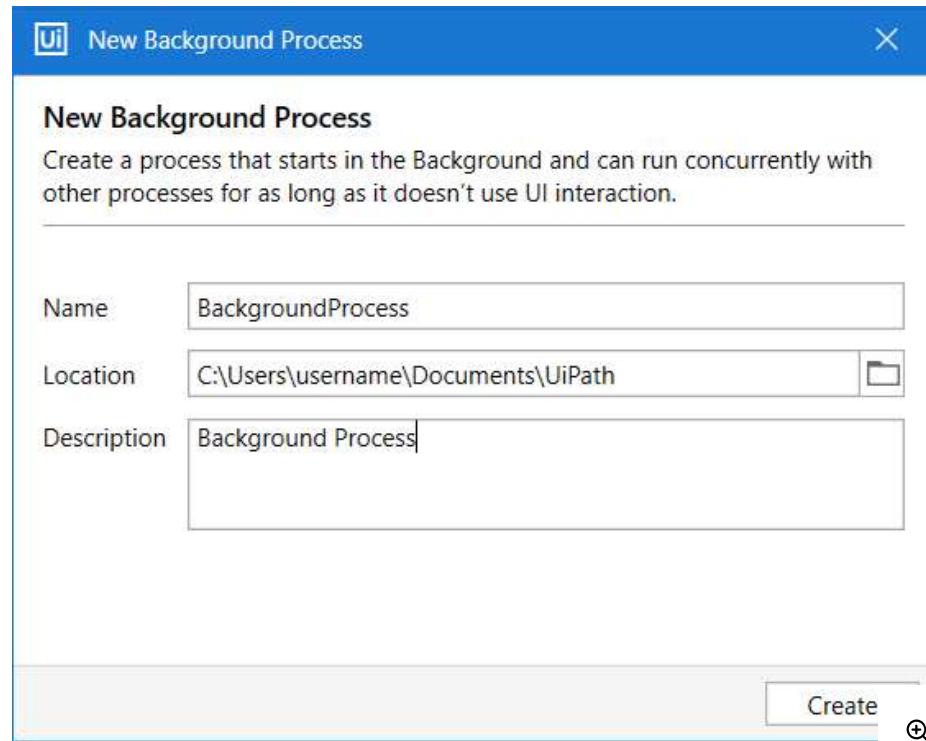
## Background Process

The **Background Process** is a template for creating processes that can run in parallel on the same Robot, together with one foreground process. For this reason, background processes must not contain activities that require user interaction.

The default dependencies of a background process template are: [UiPath.Excel.Activities](#), [UiPath.Mail.Activities](#), [UiPath.System.Activities](#), and [UiPath.WebAPI.Activities](#).

Background processes must not make use of interactive activities, like [Click](#) or [Type Into](#) found in the [UiPath.UIAutomation.Activities](#) package. Check out the [Background Process Automation](#) page to read more about how the Robot handles such processes.

In Studio, go to the Home Backstage view and select **Background Process**. The **New Background Process** window opens. Click **Create** after filling in the fields.



A process may be turned into a **Background Process** as long as it does not contain activities with UI interaction. Go to the [Project Settings](#) window and set the **Starts in Background** toggle to **Yes**.

Consequently, a **Background Process** may be turned into a foreground one by switching the same toggle to **No**.

The screenshot shows the 'Project Settings' dialog box for a 'BackgroundProcess'. The left sidebar lists 'General', 'Test Manager Configuration', 'Workflow Analyzer', 'Activities Settings', and 'System'. The 'General' tab is selected. The main area contains the following settings:

- Name:** BackgroundProcess
- Description:** Background Process
- Disable Pause:** No (checkbox)
- Starts in Background:** No (checkbox) (The cursor is hovering over this checkbox.)
- Supports Persistence:** No (checkbox)

At the bottom right are 'OK' and 'Cancel' buttons.

Before publishing a background process, make sure it does not contain any interactive activities. Using such activities in background processes, while running at the same time with foreground processes may cause unexpected results.

Please note that a **Background Process** runs in Session 0 when started from Orchestrator on an Unattended Robot.

For a more detailed description on how Background Processes work, see the [Background Process Automation](#) page, which provides more information on this subject.



Default Theme

English

# UI Automation Activities

## TABLE OF CONTENTS

### [Use Foreground](#)

Properties

In UiPath Assistant

Queue Mechanism

## Use Foreground

`UiPath.Core.Activities.UseForegroundScope`

Moves the current background process into the foreground, executing all the activities it contains. After the execution is complete, the process is moved back into the background.

## Properties

### Common

- **ContinueOnError** - Specifies if the automation should continue even when the activity throws an error. This field only supports Boolean values (True, False). The default value is False. As a result, if the field is blank and an error is thrown, the execution of the project stops. If the value is set to True, the execution of the project continues regardless of any error.

 **NOTE:**

If this activity is included in Try Catch and the value of the **ContinueOnError** property is True, no error is caught when the project is executed.

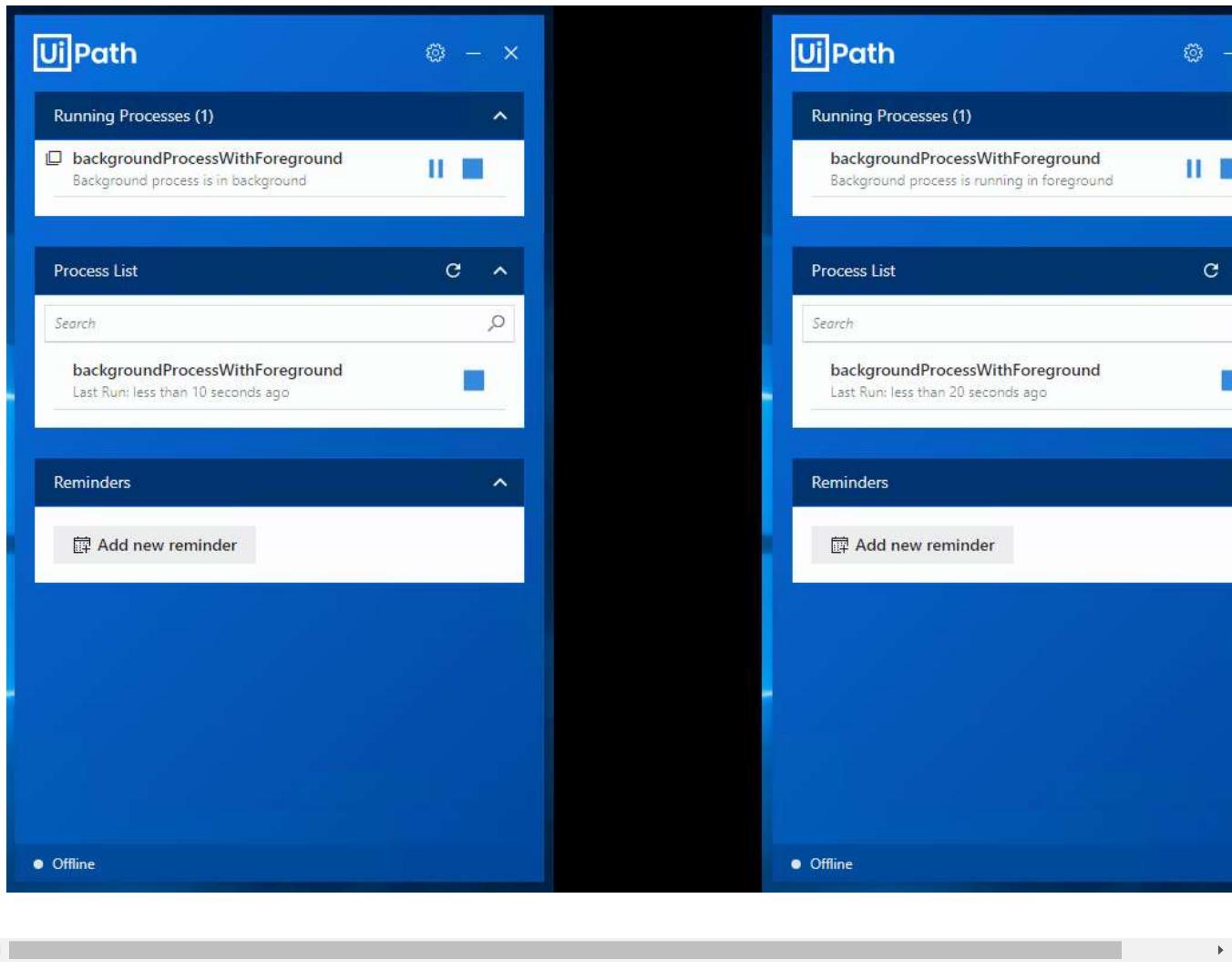
- **DisplayName** - The display name of the activity.
- **Wait for foreground** - The maximum amount of time the process should wait in order to move into the foreground. If left empty, it will wait for an infinite amount of time. If the process does not move into the foreground in the specified amount of time, an `InvalidOperationException` error is thrown. By default, this field is empty.

### Misc

- **Private** - If selected, the values of variables and arguments are no longer logged at Verbose level.

## In UiPath Assistant

When a process is running in the background it displays the **background icon** in **UiPath Assistant**. When it acquires the foreground, the **background icon** disappears:



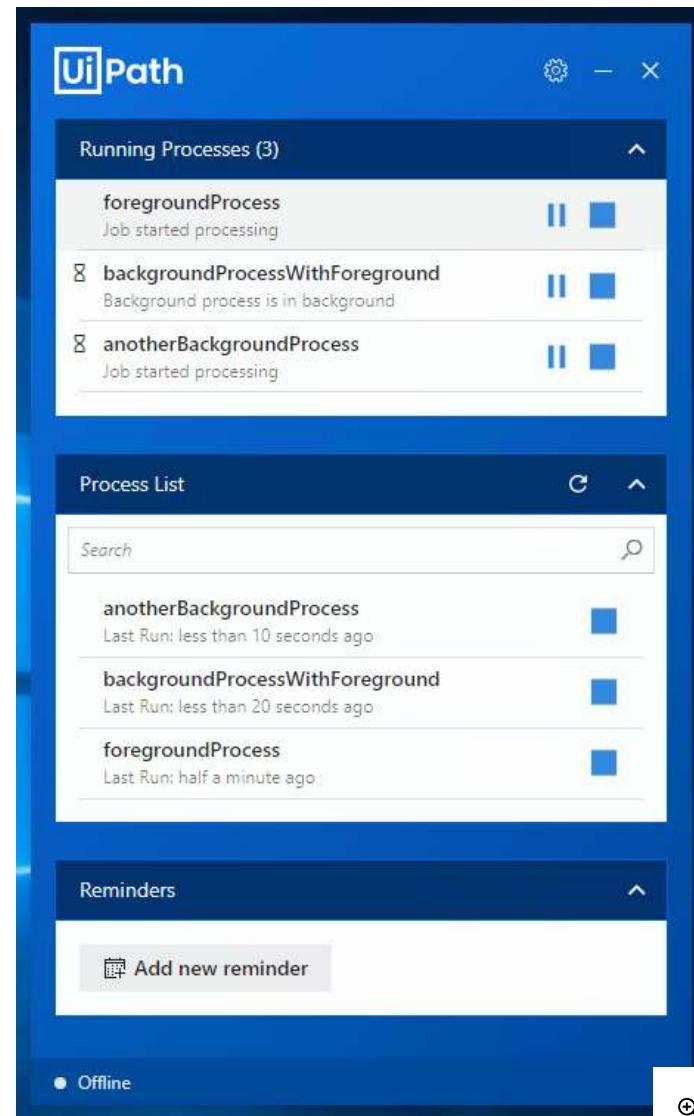
## Queue Mechanism

If you start a background process with a foreground scope and another entity is running in the foreground (either a process or another foreground scope), the background process goes into a queue and waits (determined by the **Wait for foreground** property) to acquire foreground.

There is no limit to the items in this queue.

Only background processes with a foreground scope make use of this queue, not standalone foreground processes. This means that a foreground process fails if started while you are running either another foreground process or a background process with a foreground scope.

Background processes that are waiting in the queue to acquire foreground will display an hourglass icon:



Default Theme

English

# Robot User Guide

RELEASE: 2021.10 ▾

## TABLE OF CONTENTS

### [Picture In Picture](#)

Controls

Marking a process as PiP ready

Using PiP for Invoke activities

Enabling PiP

Known Issues and Limitations

Microsoft Office automation

Using web browser in PIP sessions

PiP Requires login every time

Workflow takes a long time to start in PiP

Windows policies

VPN client not working in PiP

Cisco Anyconnect

Zscaler

Pulse secure

Palo Alto Global Protect

PIP and Windows Servers

PIP and Other Virtualized Environments

# Picture In Picture

The Picture-in-Picture feature allows you to run attended processes in collaboration with the Robot. A process started in the Picture-in-Picture mode runs in an isolated Windows session, thus allowing you to use the machine while the process is running.

---

You can start a process in Picture-in-Picture mode either from the **Debug** tab in Studio, from StudioX, or directly from the UiPath Assistant from the Contextual Menu  of a process.



#### *ⓘ* NOTE:

The default timeout to start a process in PiP session is 60 seconds. If the login in the Picture-in-Picture session takes more than that, a timeout error is thrown. This default timeout can be changed using `UIPATH_SESSION_TIMEOUT` environment variable on the machine. When using the Robot in Service-Mode, make sure to set the `UIPATH_SESSION_TIMEOUT` variable as a system environment variable and restart the Robot Service.

Once a process is started in Picture-in-Picture mode, a preview window shows up on your desktop, providing real-time feedback from the process execution. The display window can be resized, moved around, placed in full screen, or put on top of other windows. You can exit the Picture-in-Picture mode at any moment by right-clicking the Picture-in-Picture Windows Taskbar entry, and selecting **Close Window** or simply closing the window. A confirmation dialog appears and choosing to close the PiP window stops the running process.

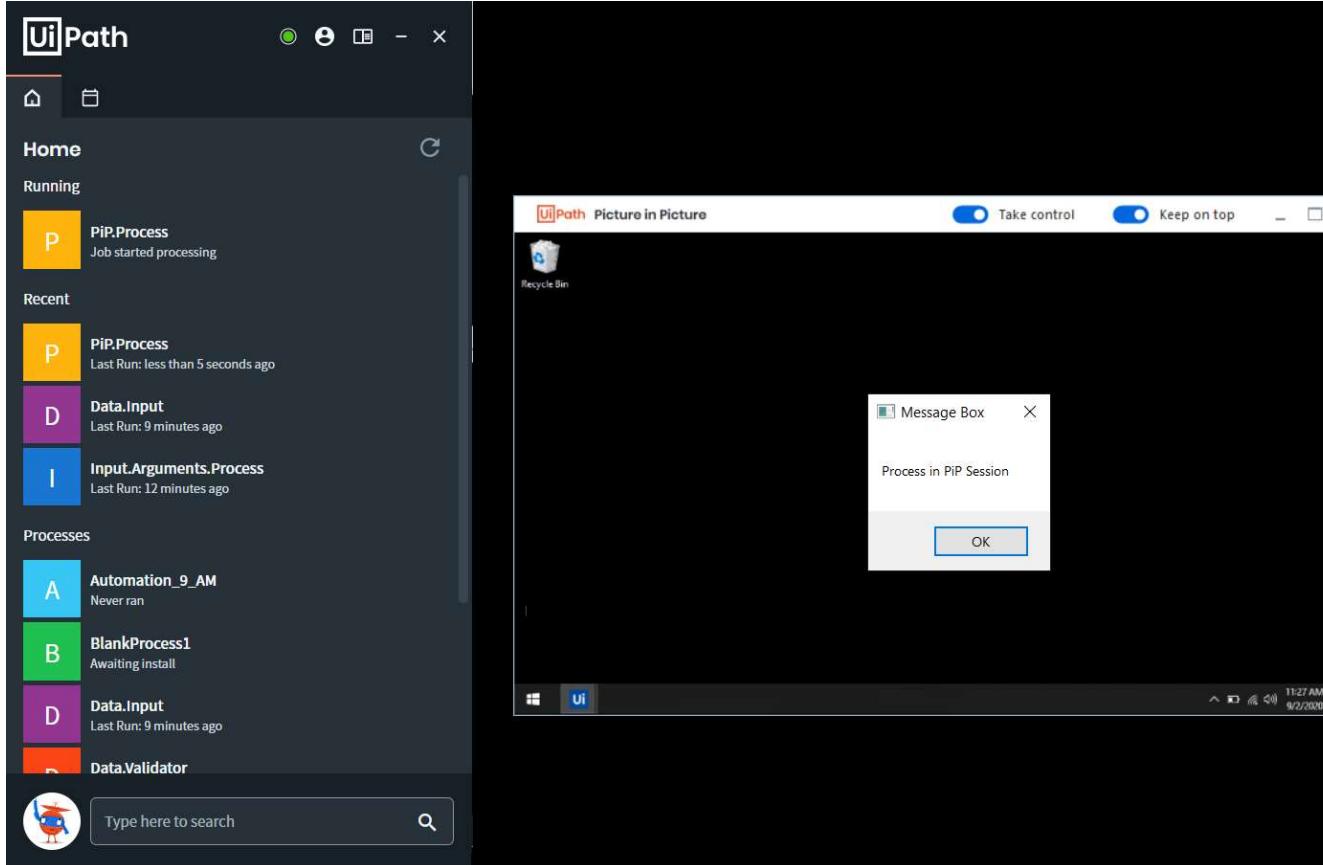
#### *ⓘ* NOTE:

Admin rights are needed to enable the Picture-in-Picture functionality on the machine. This is needed only the first time Picture-in-Picture is used, afterwards, the actual process can be started in Picture-in-Picture without elevated privileges.

## Controls

The following controls are available for the PiP window:

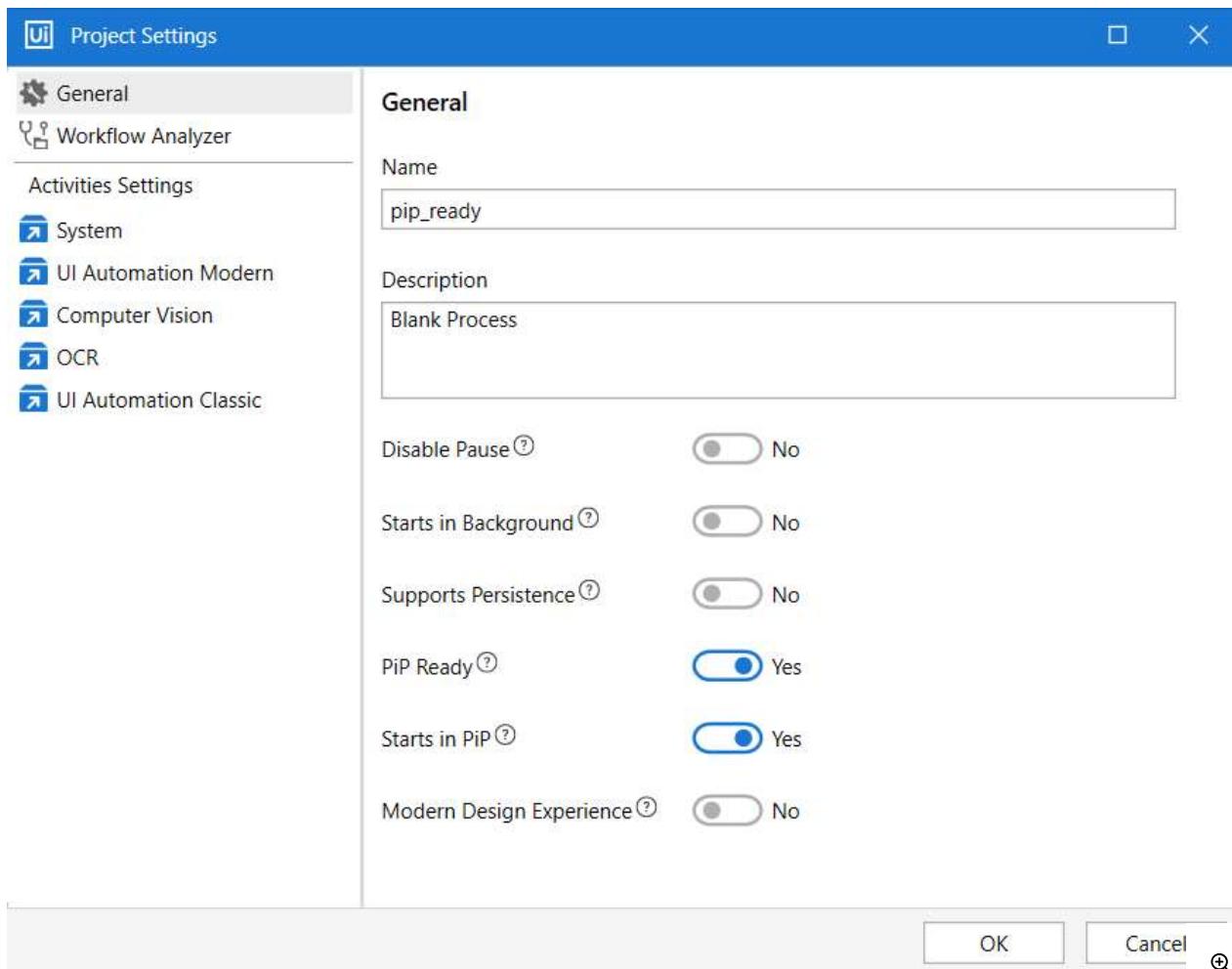
- Take control - Enable this to take control of the PiP session, if this is disabled, your mouse and keyboard only work in the main session.
- Keep on top - Enable this to keep the PiP window on top of other applications even when it's out of focus.
- Minimize - Minimizes the PiP window to the main session taskbar without interrupting the process.
- Maximize - Maximizes the PiP window.
- Close - Closes the PiP window, stops any running process and logs off the PiP session.



When you first start a Picture-in-Picture session, a prompt asks for your Windows credentials. Please note that if you restart the machine you are not asked to provide the credentials again.

## Marking a process as PiP ready

In the **Project Settings** section in Studio, a process can be marked as **PiP Ready**, meaning that it has been tested and can be safely run in a PiP Session. You can also set a process to start by default in a Picture-in-Picture session.



## Using PiP for Invoke activities

Invoke Activities such as Invoke Process, Invoke Workflow and Run Parallel Process have the option to choose where to start the new process.

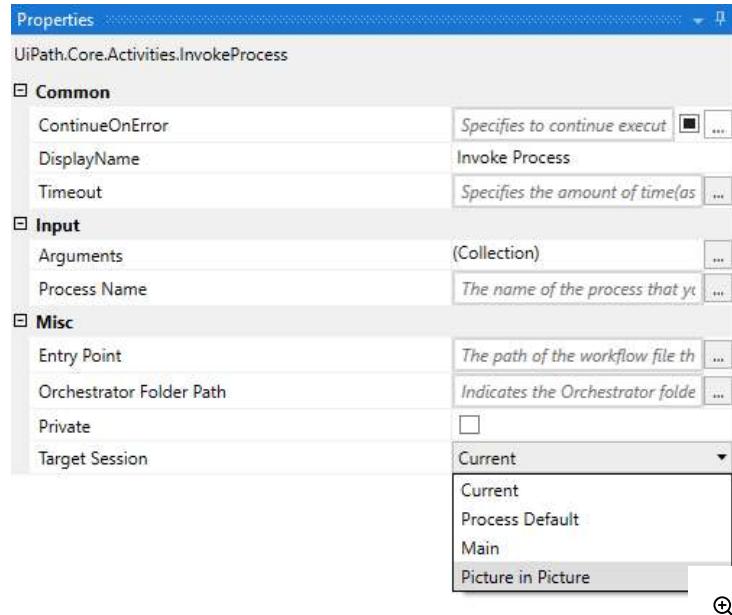
This can be set from the Properties tab of the activity in the **Misc > Target Session** inside Studio.

**NOTE:**

The **Target Session** property can only be modified from **Studio**. Projects developed in **StudioX** need to be opened in **Studio** to alter this property.

The options are:

- Current - The child process opens in the same session as the parent one
- Process Default - The child process uses the Process Settings
- Main - The child process starts in the main session regardless to where the parent process runs
- Picture-in-Picture - The child process starts in the Picture-in-Picture session regardless to where the parent process runs



## Enabling PiP

The Picture-in-Picture functionality of the machine can be either enabled through command-line, or manually when starting the PiP session for the first time on the machine.

Method	Command	Description
Manually		The first time you start a Picture-in-Picture session from either Studio or Assistant, you are prompted to enable the PiP functionality on the machine. This requires administrator rights
Command-line	UiRobot.exe PiP	<p>Allows you to enable or disable the Picture-in-Picture functionality on the machine. This setting is applied on the local machine and affects all users and is <b>used for modifying existing installations</b>.</p> <p>It can have the following parameters:</p> <ul style="list-style-type: none"> <li>• PiP --enable</li> </ul> <p>Enables the Picture-in-Picture functionality of the machine.</p> <ul style="list-style-type: none"> <li>• PiP --disable</li> </ul> <p>Disables the Picture-in-Picture functionality of the machine.</p> <p>Example: <code>UiRobot.exe PiP --Enable</code></p> <p>Administrator rights are required to execute these commands.</p>
Command-line	UiPathStudio.msi ENABLE_PIP	<p>Allows you to enable the Picture-in-Picture functionality of the machine <b>during the UiPath command-line installation</b>.</p> <p>To enable it, use the following parameter:</p> <ul style="list-style-type: none"> <li>• ENABLE_PIP=1</li> </ul> <p>Example:</p> <pre>UiPathStudio.msi ADDLOCAL=DesktopFeature,Studio,Robot,RegisterService,Packages ENABLE_PIP=1</pre>

## Known Issues and Limitations

There are a few things to consider when using the Picture-in-Picture feature:

<https://docs.uipath.com/robot/standalone/2021.10/user-guide/picture-in-picture>

- If you are using a PIN to log into the main Windows session, you are asked for your credentials every time you start a Picture-in-Picture session.
- When the Picture-in-Picture session is opened, start-up programs open in the PiP session as well. Because of this, some settings for peripheral devices can be reset to their default values (such as lighting settings for keyboard and mouse).
- If you enable the Remote Desktop Session when the prompt appears while running a PiP Process, you need to log out and log back in the main Windows Session for the changes to take effect.
- The machine cannot be restarted or shut down while the PiP session is opened as the PiP session needs to be closed beforehand.
- Due to Operating System limitations for running Picture-in-Picture, Home Editions of Windows 8 and 10 are not supported.
- The clipboard is shared between the PiP session and the main session.
- Run as administrator cannot be used in the PiP session.
- Only one Picture-in-Picture session can be started at a time.
- For a user to start a PiP session, it must be granted Allow Log On Locally permissions.

## Microsoft Office automation

Automations that use Microsoft Office resources do not run successfully in Picture-in-Picture if the resources are already open in the main session. In order to make sure that automations run smoothly in PiP, you can do the following:

- Close the resource used by Microsoft Office applications from the main session so they can be opened in the PiP session.
- Use an `InvokeIsolatedWorkflow` activity to invoke the part of the automation using Microsoft Office and set its `Target Session` to Picture-in-Picture from Studio.

 **NOTE:**

Microsoft Outlook is not affected by the limitation described above.

## Using web browser in PIP sessions

The browser data from a Picture-in-Picture session is saved on the main session by default. If there is an open Google Chrome or Microsoft Edge instance on the main session, it has to use another user profile in the PiP. This is done automatically by the Open Browser activity. We cannot have a specific browser (let's say Chrome) be open with the same user profile both in the PiP session and the Main session at the same time.

However, the mode and location of the browser data can be configured from the `Open Browser` activity properties.

Setting the `UserDataAdapterMode` property to `Automatic` allows the browser to use separate user data folders in the main and PiP sessions.

Please note that if you clear the user data from the `%LocalAppData%\UiPath\PIP Browser Profiles` folder in this mode, the corresponding browser extension needs to be enabled again.

In case you need to use data from the main session (such as cookies or saved passwords), consider setting the `UserDataAdapterMode` property to `DefaultFolder`. This means that both the main and PiP sessions use the same folder for the browser user data.

 **NOTE:**

When setting the `UserDataAdapterMode` to `DefaultFolder` the browser only works in one session at a time. If the browser is opened in the main session, it does not work in the PiP session. This is because the same browser profile cannot be used in two simultaneously sessions.

 **NOTE:**

`Target Session` and `UserDataAdapterMode` are properties that can only be modified in **Studio**. Projects developed in **StudioX** need to be opened in **Studio** to alter these properties.

Setting the `UserDataAdapterMode` property to `CustomFolder` allows you to specify different user data folders for the main and PiP sessions.

## PiP Requires login every time

Based on your environment, certain Windows policies might cause the PiP session to request a login every time it starts.

This also happens when Windows Business Hello PIN is used. PIN authentication only works the first time a PiP session is spawned. After that, the PiP session can be logged in only using username and password. As this might not be a best case scenario for some organizations, we are currently looking to correct this behavior in the future.

## Workflow takes a long time to start in PiP

When launching a process in PiP for the first time, it takes longer than usual until the actual execution starts. This happens because the PiP session has to start all its Windows processes and start-up programs.

**Recommendation:** Launch a PiP session when starting the machine and keep it open throughout the day. This uses less resources than launching a new PiP session for every process.

## Windows policies

Some Windows policies such as AllowLogOnLocally can restrict the PiP session from starting. The user trying to start a PiP session must be in a group that has the needed permissions, or must be explicitly given the permission.

 **NOTE:**

We are currently investigating to find other policies that can affect the PiP functionality.

## VPN client not working in PiP

When VPN clients are used in conjunction with PiP, there are some situations in which conflicts may occur. For example, if the VPN clients are set to start when the user logs in, when PiP starts, another instance of the VPN client is started. This causes a conflict between the two sessions, since the VPN client is set to run a single instance per user.

To resolve these scenarios, we have compiled a list of the most common VPN providers with their particularities, plus resolutions for the known issues that may occur.

### Cisco Anyconnect

#### Observed Behavior

When the Cisco Anyconnect client is running on the user machine and a PiP session is launched, another Cisco Anyconnect client is started in the PiP session.

**Cause** The Cisco VPN server is set to accept one session per user at a time. When the PiP session starts, Cisco Anyconnect disconnects the VPN in the main session and throws an error in the PiP session.

This shuts down the user's VPN connection, leaving the user unable to access services that require a VPN connection.

**Resolution** Do not set the VPN client to start automatically at Windows start-up. This stops the VPN client from starting a new connection when the PiP session starts and tunnels the PiP traffic through the main Windows session.

### Zscaler

#### Observed Behavior

When the PiP session is started, another Zscaler client is launched in the PiP session. This causes the Zscaler client to disappear from the main Windows session. Everything works as expected until the user closes the PiP session. When the PiP session is closed, the Zscaler client remains in a limbo state and the user must sign out and sign in again, or restart the machine to open the Zscaler.

**Cause** We are currently investigating this behavior with Zscaler to find the cause. **Resolution** We are currently investigating this behavior with Zscaler to find a resolution.

### Pulse secure

#### Observed Behavior

When a PiP session is started, the user is disconnected from the VPN.

**Cause** The Pulse secure client cannot handle two Windows sessions for the same user. **Resolution** We recommend opening a ticket with Pulse Secure team.

### Palo Alto Global Protect

When PiP starts, the GUI shows the user disconnected from the VPN in both sessions. But the PiP session is still connected to the VPN.

**Resolution** We recommend opening a ticket with Palo Alto team.

## PIP and Windows Servers

In a scenario where multiple users are connected at the same time to a Windows Server, only one PIP session can be launched on the machine. This means only one PiP session can be opened on a single machine, regardless of which user or session type was used to open the PiP session.

## PIP and Other Virtualized Environments

On other virtualized environments such as AppV or Citrix XenApps, PIP functionality is not available.



Default Theme

English



# Robot User Guide

RELEASE: 2023.4

## TABLE OF CONTENTS

### [Picture in Picture](#)

[Setting the PiP type](#)

[Marking a process as PiP ready](#)

[Using PiP for Invoke activities](#)

## Picture in Picture

Picture-in-Picture allows you to run attended automations without having to interrupt your current activity on the machine.

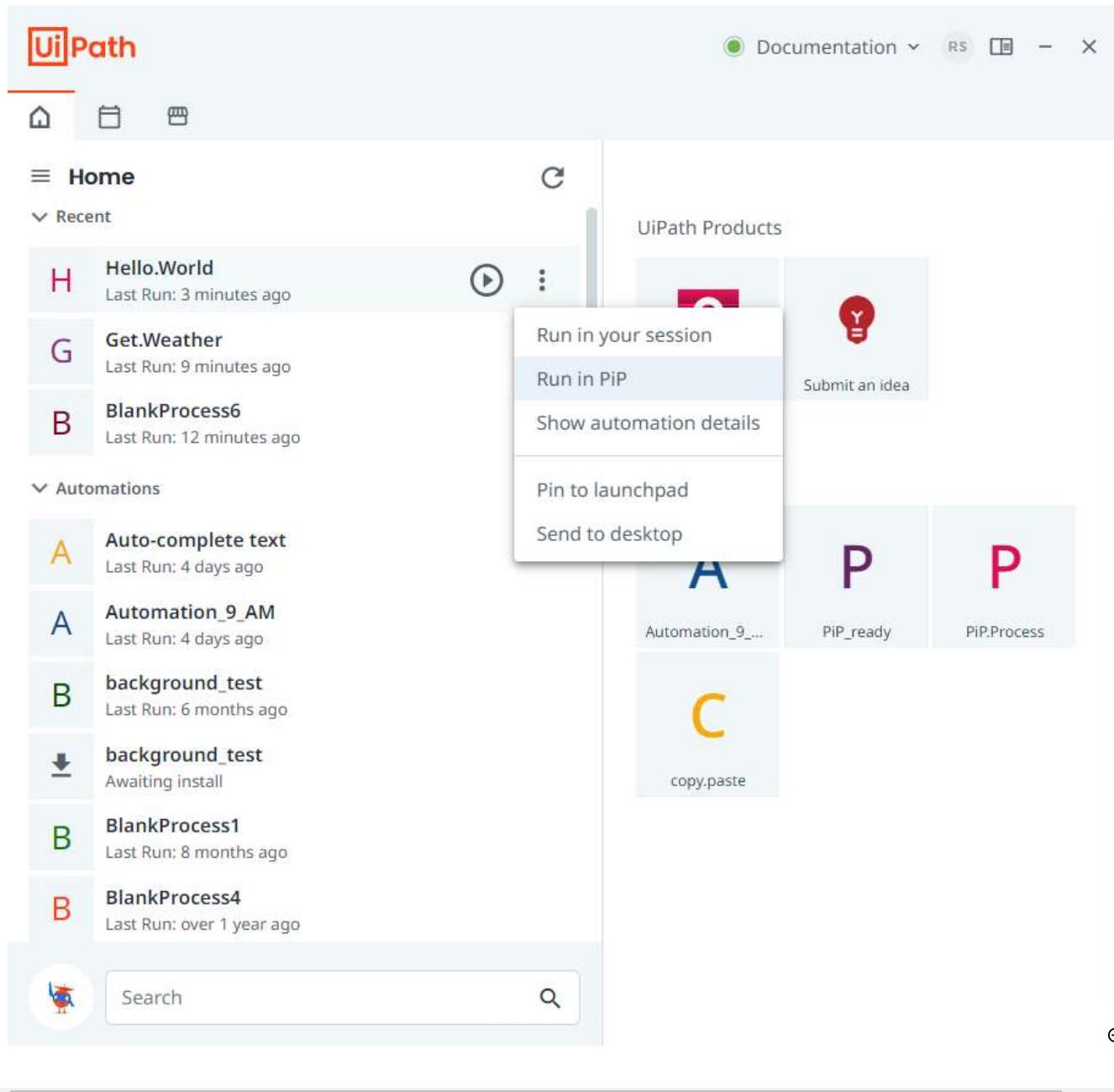
Just as you can consume media while using the computer with the help of Picture in Picture, the work you have can be separated as well.

While the Robot works in PiP, your machine is free and you can access your files, modify documents, send e-mails, answer phone calls, and other duties that can only be completed by you.

There are two versions of Picture in Picture, both isolating the automation from your work, and are based on the following technologies:

- **Child Session** - Processes run in a separate Windows session on the machine.
- **Virtual Desktop** - Processes run in the same Windows session but on a virtual desktop.

A PiP process is started from either the **Debug** tab in Studio, from StudioX, or from the UiPath Assistant.

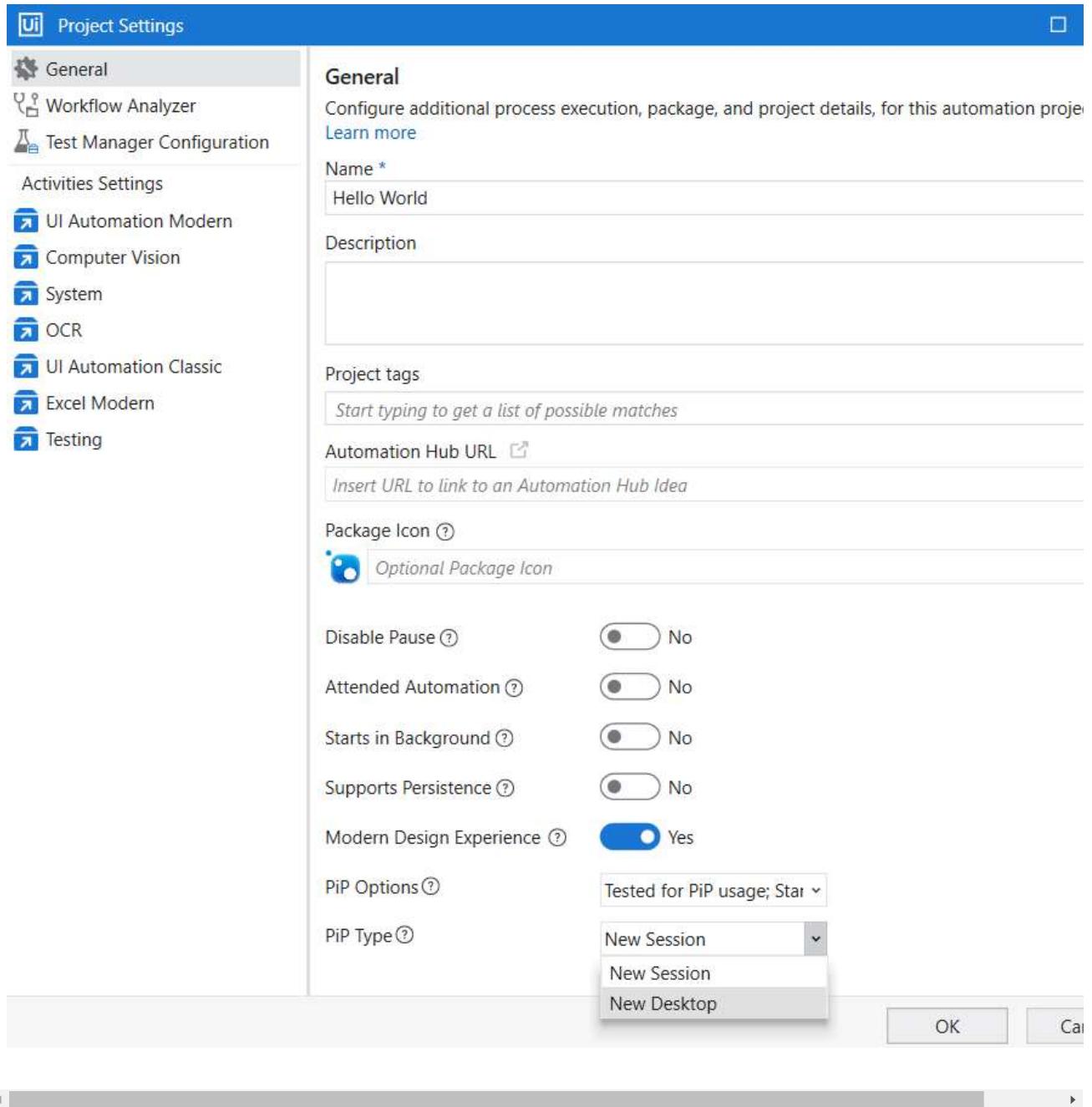


## Setting the PiP type

The PiP technology used by your automations is set from the **PiP Type** menu in the **Project Settings** in Studio. By default, this is set to New Session.

The following options are available:

- **PiP Type:**
  - **New Session** - When the automation is run in PiP, the child session technology is used.
  - **New Desktop** - When the automation is run in PiP, the virtual desktop technology is used.



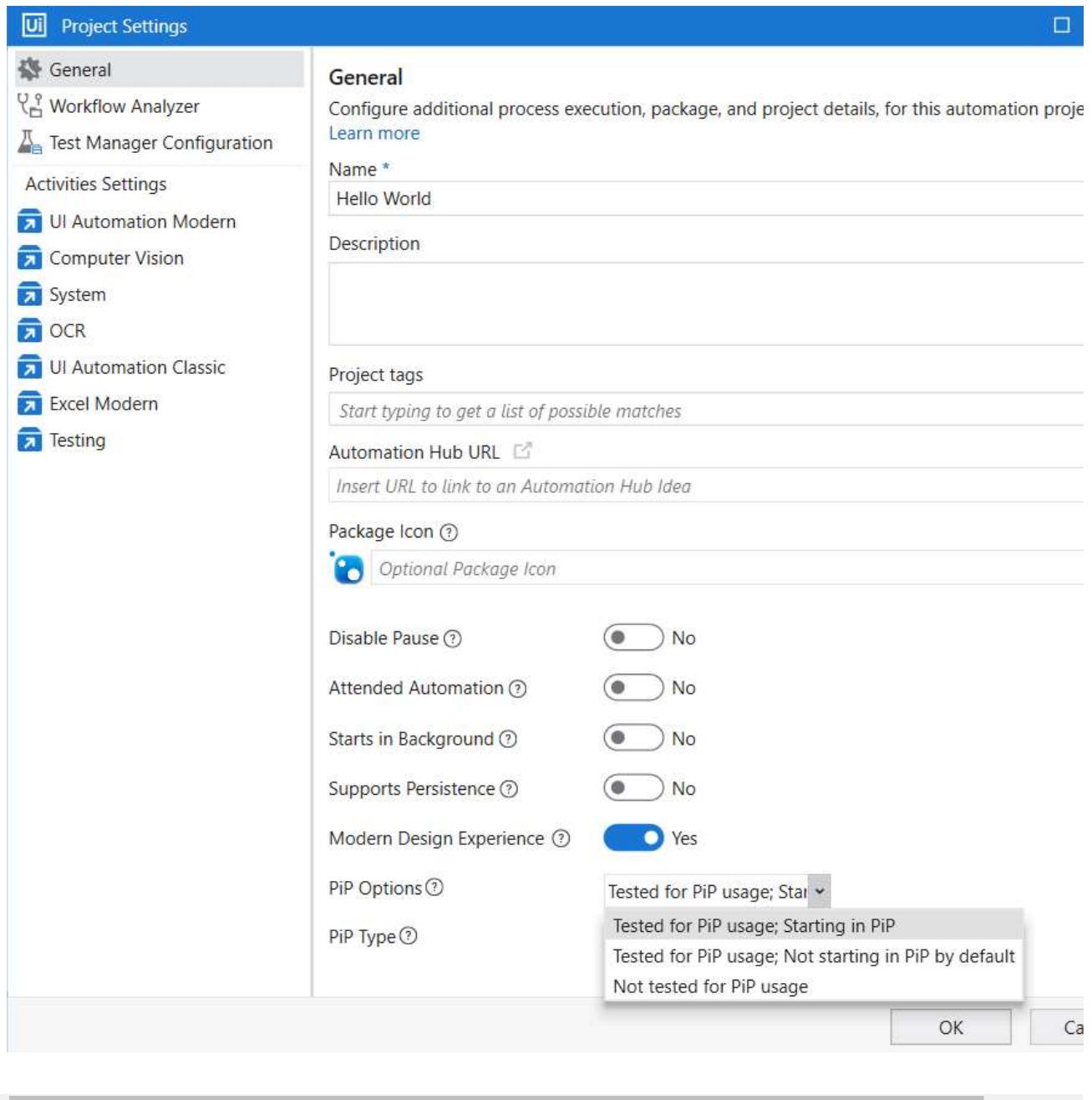
## Marking a process as PiP ready

If a process has been tested and can be safely run in PiP, you can mark it as such from the [Project Settings](#) in Studio.

From the same menu, you can also set it to start by default in Picture-in-Picture, and choose the PiP technology you want to use:

- **PiP Options:**

- **Tested for PiP Usage; Starting in PiP** - The automation has been approved to run in PiP mode. When run, it starts in PiP by default.
- **Tested for PiP Usage; Not starting in PiP by default** - The automation has been approved to run in PiP mode. When run, it starts in the main session by default.
- **Not tested for PiP usage** - The automation has not been approved to run in PiP mode. When run, it starts in the main session by default.

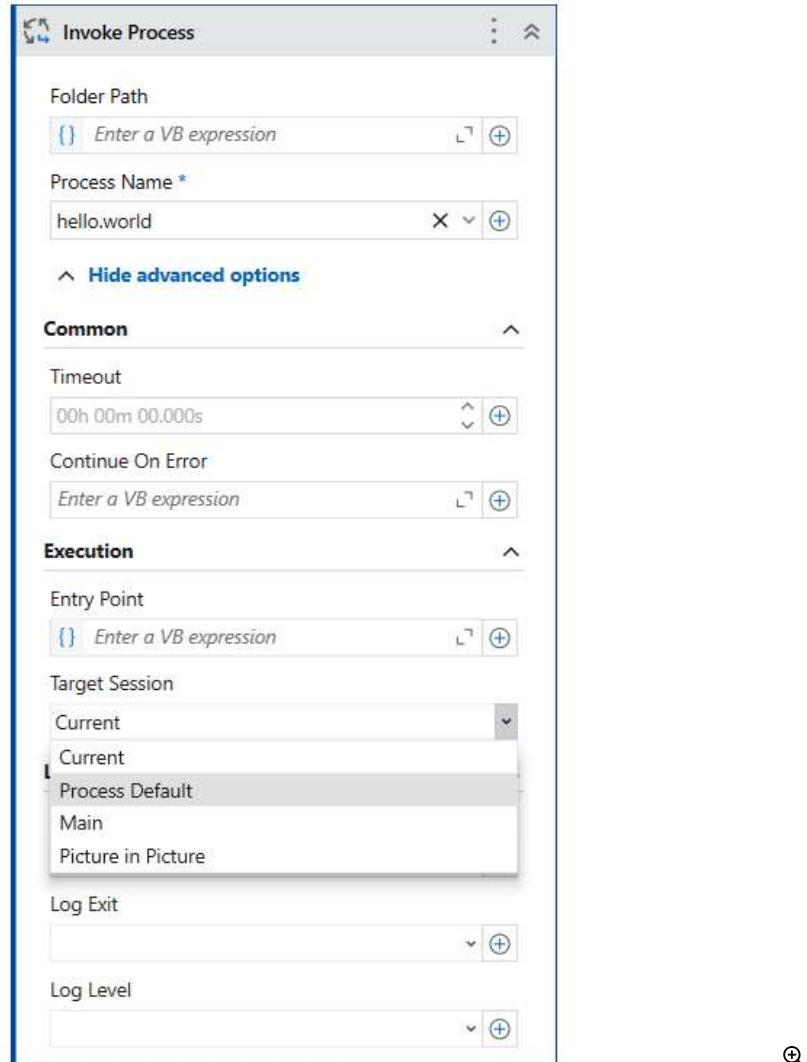


## Using PiP for Invoke activities

When using Invoke Activities such as [Invoke Process](#), [Invoke Workflow File](#), and [Run Parallel Process](#), you can choose the session in which they run.

This is done by setting the **Target Session** property of the activity in Studio using the following values:

- Current - The child process opens in the same session or desktop the user is in.
- Process Default - The child process uses the configuration set in the **Process Settings**.
- Main - The child process starts in the user session regardless of where the parent process runs.
- Picture-in-Picture - The child process starts in Picture-in-Picture regardless of where the parent process runs.



**NOTE:**

The Target Session property can **only** be modified from **Studio**.

To alter this property for projects developed in **StudioX**, you must open them in **Studio**.



Default Theme

English



# Studio User Guide

RELEASE:

2023.4



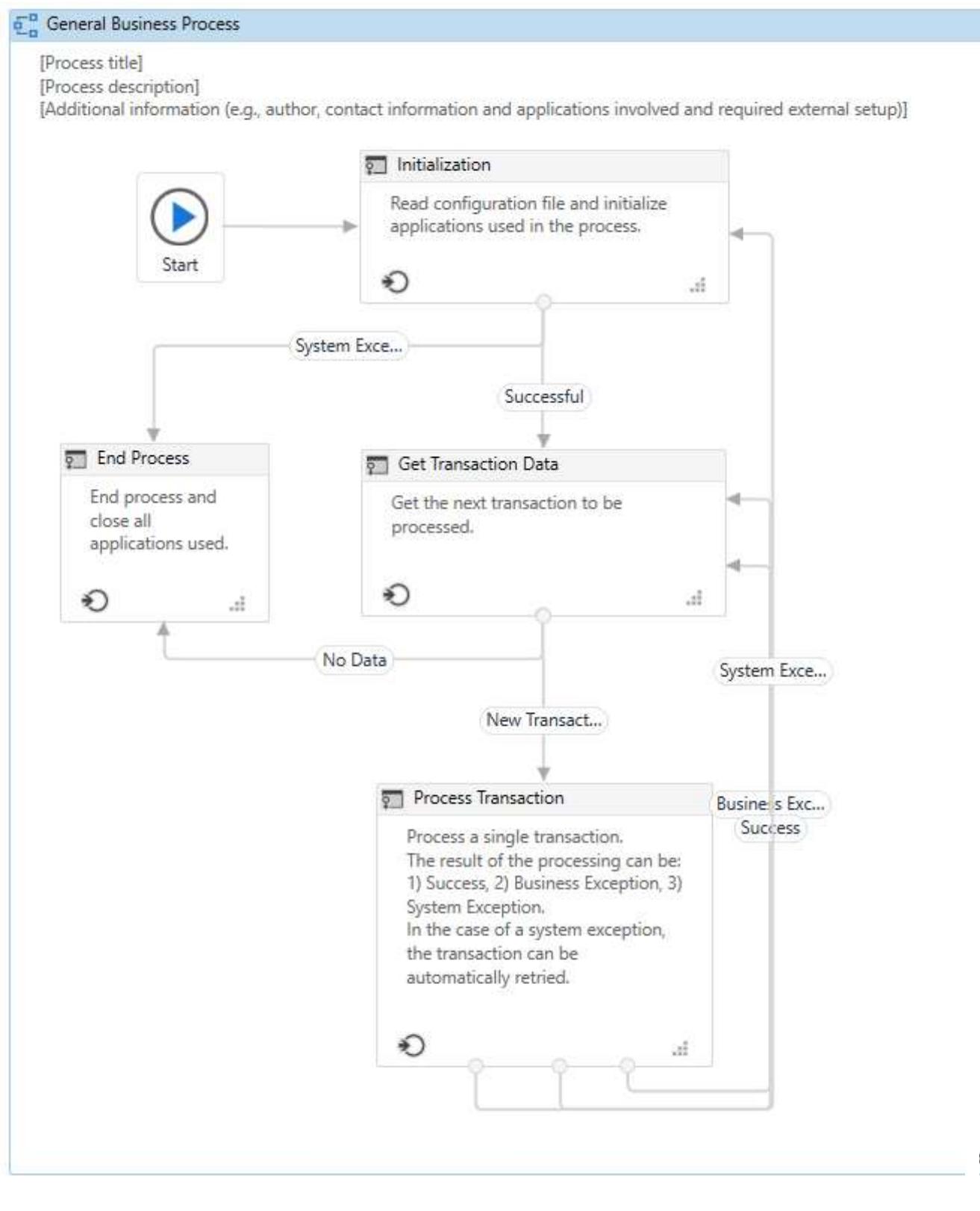
## TABLE OF CONTENTS

### [Robotic Enterprise Framework](#)

## Robotic Enterprise Framework

**Robotic Enterprise Framework** is a project template based on **State Machines**. It is created to fit all of the best practices regarding logging, exception handling, application initialization, and others, being ready to tackle a complex business scenario.

The template contains several pre-made State containers for initializing applications, retrieving input data, processing it and ending the transaction. All of these States are connected through multiple Transitions which cover almost every need in a standard automation scenario. There are also multiple **invoked workflows**, each handling particular aspects of the project.



The default dependencies in a **Robotic Enterprise Framework** project are: **UiPath.Excel.Activities**, **UiPath.System.Activities**, and **UiPath.UIAutomation.Activities**.

The template comes with detailed documentation and examples in the project folder.



Default Theme

English

# Robot User Guide

RELEASE:

2022.10



## TABLE OF CONTENTS

### [UiPath.Settings File Description](#)

## UiPath.Settings File Description

The `UiPath.Settings` file contains all the necessary details regarding how the Robot performs processes. You can modify these settings by directly editing the file and the corresponding fields. On the other hand, you can modify them via Orchestrator. This is done from [the Settings tab](#).

 **NOTE:**

Please note that local system and administrator rights are required to modify these settings. You should restart the UiPath Robot service (admin rights are required to do this), and close the Robot tray for changes to take effect.

The `UiPath.Settings` file is stored in the `%localappdata%\UiPath\` folder when the Robot is deployed in user mode, and `%programdata%\UiPath\` when the Robot is deployed in service mode. It is created the first time the UiPath Robot service starts. It contains the following parameters:

Parameter	Description
<code>NuGetApiKey</code>	The API key of the NuGet feed.

Parameter	Description
	<p><b>When not connected to Orchestrator</b>, if you are using a local feed, it does not require an API key. If you use a private MyGet feed, please note that this parameter is required.</p> <p><b>When connected to Orchestrator</b>, this value is not taken into account.</p>
<b>NuGetServerUrl</b>	<p>The location where projects are pushed and from where they are retrieved. This can be either a local feed, such as a file system path, or a web feed that uses the NuGet protocol (NuGet, MyGet etc.).</p> <p><b>When not connected to Orchestrator</b>, the default value is %ProgramData%\UiPath\packages (user mode) or %. If you use a private MyGet feed, please use the URL provided under <b>Your pre-authenticated V2 URL (no basic authentication)</b>. Please note that this is not a free service from MyGet.</p> <p><b>When connected to Orchestrator</b>, this value is not taken into account.</p>
<b>ActivitiesFeed</b>	<p>The address where the activities are stored. This field is only populated if you are <b>not</b> connected to Orchestrator. By default, the value is ~/NuGetPackages/Activities. If you connect to Orchestrator, this field is not taken into account. Find out more about the activities feeds <a href="#">on this page</a>, as well as where packages are installed based on the <a href="#">Robot deployment type</a>.</p>
<b>UiPathServerUrl</b>	<p>The address of your instance of Orchestrator.</p> <p><b>When not connected to Orchestrator</b>, this parameter is empty.</p> <p><b>When connected to Orchestrator</b>, it is automatically filled in with the URL you provided in the <a href="#">Orchestrator Settings window</a>.</p>
<b>TracingLevel</b>	<p>The level at which the Robot should log information. The following options are available:</p> <p><b>Verbose, Trace, Information, Warning, Error, Critical</b> and <b>Off</b>.</p> <p>This can also be changed from the <a href="#">Orchestrator Settings window</a>, from the <b>Level</b> drop-down list.</p>
<b>LowLevelTracing</b>	<p>Indicates whether or not to log information at a trace level, to help you with troubleshooting crashes and errors. By default, this parameter is set to <b>false</b>. To start low level tracing, set the</p>

Parameter	Description
	value of this parameter to true. You can also change this value from <a href="#">the command line</a> .
<b>SecureLicenseKey</b>	The Machine Key that is used to connect to Orchestrator. The key is encrypted in the UiPath.settings file using DPAPI. This value can also be filled in and modified from the <b>Orchestrator Settings window</b> . If not connected to Orchestrator, this parameter should be empty.
<b>LoginToConsole</b>	Enables the Robot to connect to the console session of the machine where it is installed. Additionally, it indicates if you can connect multiple Robots to Orchestrator using multiple users ( <a href="#">High-Density Robots</a> ) or not. By default, the value is true. To enable High-Density Robots, set the value to false.
<b>ResolutionWidth</b>	The machine's display resolution width. This option is set to 0 by default, which means that it automatically retrieves and uses the detected resolution width. You can use a custom value, as long as it is supported by the workstation.
<b>ResolutionHeight</b>	The machine's display resolution height. This option is set to 0 by default, which means that it automatically retrieves and uses the detected resolution height. You can use a custom value, as long as it is supported by the workstation.
<b>ResolutionDepth</b>	The machine's display resolution depth. This option is set to 0 by default, which means that it automatically retrieves and uses the detected resolution depth. You can use a custom value, as long as it is supported by the workstation.
<b>FontSmoothing</b>	Enhances text recognition. If is set to <b>True</b> , the <b>Font Smoothing</b> option over RDP connections is turned on. This means that the target machine will have <b>ClearType</b> turned on.
<b>ConnectionString</b>	A string generated from Orchestrator that enables you to register your Robot to it, without providing the <b>Machine Name</b> and <b>Machine Key</b> .
<b>DisableWorkflowExecution</b>	Disables the possibility to execute processes from the command line when enabled. By default, this feature is hidden

Parameter	Description
	and disabled.
<b>webProxySettings</b>	Parameter used to configure Proxy Settings for the Robot. Find out more in the <a href="#">Redirecting Robots Through a Proxy</a> document.



Default Theme

English

# Workflow Activities

## TABLE OF CONTENTS

---

### [Set Transaction Status](#)

Description

Project compatibility

Windows, Cross-platform configuration

Windows - Legacy configuration

## Set Transaction Status

UiPath.Core.Activities.SetTransactionStatus

## Description

Sets the status of an [Orchestrator transaction](#) item to Failed or Successful. An example of how you can use this activity is available [here](#).

 **NOTE:**

- Schema definition values added to queues in Orchestrator and marked as required are automatically imported in the Dictionary Builder of the activity.
- Values marked as not being required can be added to queue items and not be enforced.

# Project compatibility

Windows - Legacy | Windows | Cross-platform

## Windows, Cross-platform configuration

- **Orchestrator Folder Path** - [The path of the folder](#) where the **TransactionItem** is located, if different from the folder where the process is running. This field only supports string values, with / as the separator to indicate subfolders. For example "Finance/Accounts Payable".

 **NOTE:**

The Orchestrator Folder Path parameter does not work for processes executed by robots in classic folders. Only robots in modern folders have the ability to pass data outside of their folder.

Relative folder paths are supported in an X-UIPATH-FolderPath-Encoded header, as follows:

-right: Path starting with / - starts from the root folder of the tree the ambient folder is part of.

-right: Path starting with . - starts from the ambient folder.

-right: Path starting with .. - starts one level up in the hierarchy of the ambient folder for each .. in the path (e.g. .../ for one level up, .../.../ for two levels up in the hierarchy).

Note that trailing slashes are not accepted.

- **Transaction Item** - The TransactionItem whose status is to be updated.
- **Status** - The status that is to be set to the TransactionItem.

### Advanced options

#### Input

- **Queue Name** - The queue where to select the schema definition from. The name is case insensitive, meaning that if in Orchestrator it was defined as "MyFirstQueue", it matches "myfirstqueue". The maximum number of characters is 50.
- **Analytics** - A collection of analytics information about the specific TransactionItem whose status is to be updated.
- **Output** - A collection of additional information about the specific TransactionItem whose status is to be updated.

## Others

- **Timeout (milliseconds)** - Specifies the amount of time (in milliseconds) to wait for the activity to run before an error is thrown. The default value is 30000 milliseconds (30 seconds).
- **Continue On Error** - Specifies to continue executing the remaining activities even if the current activity failed. This field only supports Boolean values (True, False). The default value is False.

## Transaction Error

- **Details** - Details regarding the failed Transaction. This field supports only strings and String variables. You can place any log information or other details about the failure.
- **ErrorType** - The error type that the failed Transaction has thrown. Application - a technical issue (e.g. a file that cannot be found), this type of error will retry the transaction according to the settings in Orchestrator, Business - an error regarding an external factor (e.g. an invoice that could not be paid). For more information on this choice, see the [Business Exception vs Application Exception](#) article.
- **Reason** - The reason for which the Transaction failed. This field supports only strings and String variables. You can place any short reason here (e.g. does not contain the letter F).

# Windows - Legacy configuration

## Properties panel

### Common

- **Continue On Error** - Specifies to continue executing the remaining activities even if the current activity failed. This field only supports Boolean values (True, False). The default value is False.
- **DisplayName** - The display name of the activity.
- **Timeout (milliseconds)** - Specifies the amount of time (in milliseconds) to wait for the activity to run before an error is thrown. The default value is 30000 milliseconds (30 seconds).

### Input

- **Analytics** - A collection of analytics information about the specific TransactionItem whose status is to be updated.
- **Output** - A collection of additional information about the specific TransactionItem whose status is to be updated.
- **Status** - The status that is to be set to the TransactionItem.
- **Transaction Item** - The TransactionItem whose status is to be updated.

### Misc

- **Folder Path** - [The path of the folder](#) where the **TransactionItem** is located, if different from the folder where the process is running. This field only supports string values, with / as the separator

to indicate subfolders. For example "Finance/Accounts Payable".

 **NOTE:**

The FolderPath parameter does not work for processes executed by robots in classic folders. Only robots in modern folders have the ability to pass data outside of their folder. Relative folder paths are supported in an x-UIPATH-FolderPath-Encoded header, as follows:

- right: Path starting with / - starts from the root folder of the tree the ambient folder is part of.
- right: Path starting with . - starts from the ambient folder.
- right: Path starting with .. - starts one level up in the hierarchy of the ambient folder for each .. in the path (e.g. .../ for one level up, .../.../ for two levels up in the hierarchy).

Note that trailing slashes are not accepted.

- **Private** - If selected, the values of variables and arguments are no longer logged at Verbose level.

## Transaction Error

- **Details** - Details regarding the failed Transaction. This field supports only strings and String variables. You can place any log information or other details about the failure.
- **ErrorType** - The error type that the failed Transaction has thrown. Application - a technical issue (e.g. a file that cannot be found), this type of error will retry the transaction according to the settings in Orchestrator, Business - an error regarding an external factor (e.g. an invoice that could not be paid). For more information on this choice, see the [Business Exception vs Application Exception](#) article.
- **Reason** - The reason for which the Transaction failed. This field supports only strings and String variables. You can place any short reason here (e.g. does not contain the letter F).



Default Theme

English

# Orchestrator Standalone User Guide

DELIVERY: AUTOMATION CLOUDAUTOMATION SUITESTANDALONERELEASE: 2023.4

## TABLE OF CONTENTS

### Managing Assets in Orchestrator

[Creating Assets](#)[Managing Asset Links](#)[Linking Multiple Assets to the Current Folder](#)[Linking an Asset to Multiple Folders](#)[Unlinking Assets From Folders](#)[Editing Assets](#)[Adding Tags to Assets](#)[Removing Tags From Assets](#)[Removing Assets](#)

# Managing Assets in Orchestrator

## Creating Assets

1. In the **Assets** page, click **Add**. The **Add Asset** window is displayed.
2. In the **Asset name** field, enter a name for the asset.
3. From the **Type** list, select the type of asset you want to create.
4. Add a **Description** of the asset.
5. From the **Credential Store** list, select where this asset is stored from any of your defined credential stores.
6. Use the **Global Value** toggle to indicate whether this asset has a default value (used when no specific per-account/per-account-machine value exists).
  - If enabled, in the **Value** field, type what the asset should contain. If the asset type is boolean, select **True** or **False**, and if it is a credential enter the **Username** and **Password**.
  - If disabled, you must provide at least one account-specific asset value.
7. Use the **Add robot asset value** button to create the per-account or per-account-machine asset values desired.

8. Click **Create**. The asset is created and displayed on the **Assets** page.

**ⓘ NOTE:**

Please note that once an asset is created, the asset type cannot be changed.

Field	Description
<b>Asset name</b>	A custom name for the asset to help you identify it easily. Cannot exceed 256 characters.
<b>Type</b>	The type of the asset. The following options are available: <ul style="list-style-type: none"> <li><b>Text</b> - assets that can store string values. This is the default option.</li> <li><b>Bool</b> - assets that can store a true or false value. If you select this option, the <b>Value</b> field displays two radio buttons (<b>True</b>,<b>False</b>).</li> <li><b>Integer</b> - assets that can store integers.</li> <li><b>Credential</b> - assets that can store credentials. If you select this option the <b>Value</b> field is replaced by two others: <b>Username</b> and <b>Password</b>.</li> </ul>
<b>Credential Store</b>	Available only when adding credential assets. The credential store where the asset will be saved, whether the native Orchestrator database or any of your created external stores.
<b>External Name</b>	Available only when adding credential assets to a credential store. The name of the asset to be used when communicating with the credential store.
<b>Global Value</b>	An asset with a global value asset is received by all accounts, except those with specific asset values (defined using the <b>Add robot asset value</b> button). When this option is enabled, you must fill in the global value that is received by accounts: <ul style="list-style-type: none"> <li>For <b>Bool</b> assets, two radio buttons, <b>True</b> and <b>False</b>, are displayed. By default, <b>False</b> is selected.</li> <li>For <b>Credential</b> assets the <b>Username</b> and <b>Password</b> fields are enabled. Both these fields have to be filled in.</li> <li>For text assets, this field allows for up to 1.000.000 characters.</li> </ul>
<b>Add robot asset value</b> 	Enables you to add specific asset values on a <a href="#">per account</a> or <a href="#">per account-machine</a> basis. When not using the <b>Global Value</b> option, you must configure at least one specific value. If a global value is provided, adding specific values is optional. <ul style="list-style-type: none"> <li>For <b>Bool</b> assets, two radio buttons, <b>True</b> and <b>False</b>, are displayed. By default, <b>False</b> is selected.</li> <li>For <b>Credential</b> assets the <b>Username</b> and <b>Password</b> fields are enabled. Both these fields have to be filled in.</li> <li>For text assets, this field allows for up to 1.000.000 characters.</li> </ul>

## Managing Asset Links

Sharing assets between folders enables launching jobs in multiple folders without redesigning your workflows in Studio when the underlying processes are targeting the same asset. Linking an asset to a folder makes the asset and all asset-associated objects, such as asset items, available in that folder.

**⚠️ IMPORTANT:**

You can only share assets with global values and no specific value per account. You cannot share assets with values per account between folders.

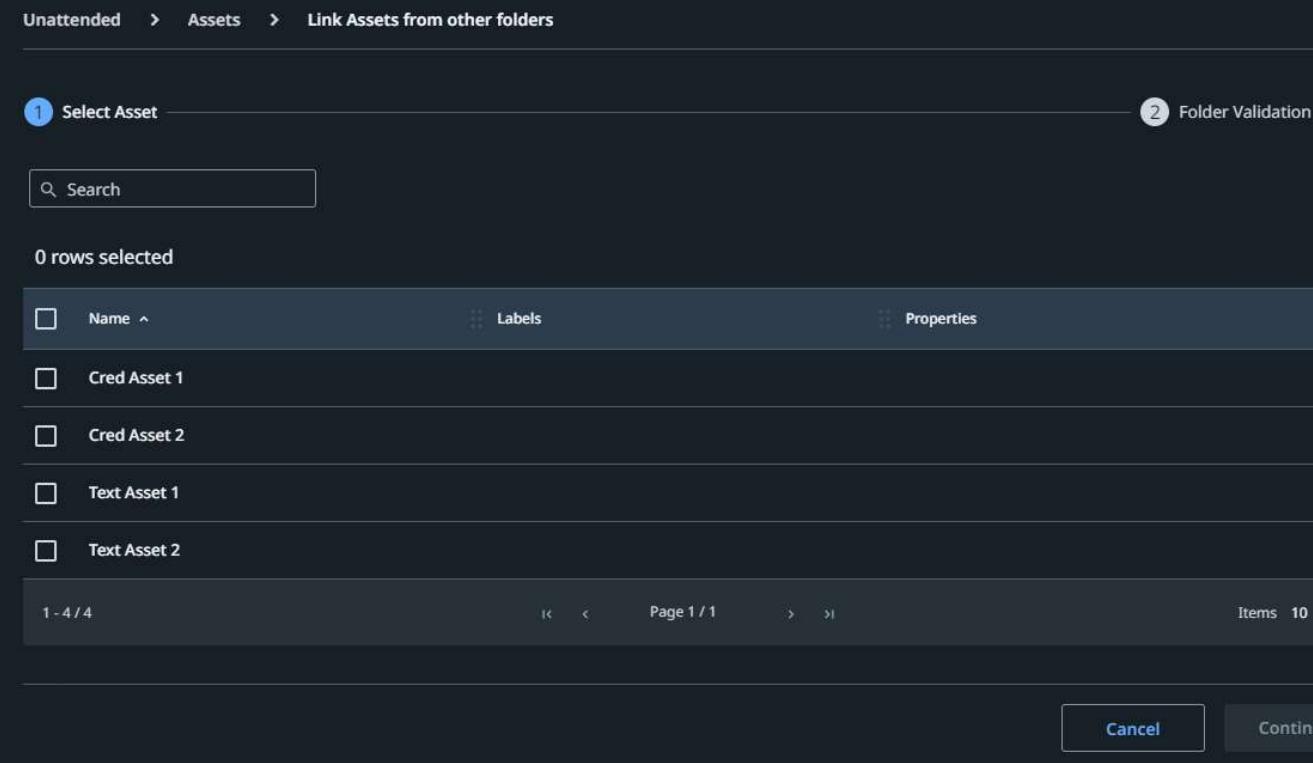
**ⓘ NOTE:**

An asset linked to multiple folders is marked using the  icon. If the icon is not present, then the current folder is the only folder the asset resides in. Deleting it here completely removes the asset from Orchestrator.

You need the **Assets - Create** permission in the folders where you want to add the asset (target folders) and **Assets - View** in the folder where the asset currently resides (original folder). If you have **Assets - Edit** in the target folder, you also require **Assets - Edit** in the original folder.

## Linking Multiple Assets to the Current Folder

1. In the folder you want to link an asset to, on the **Assets** page, click **Add**. Three buttons are displayed allowing you to add an asset, link assets from other folders, or hide the options.
2. Click **Link from other folders**. The **Link Assets** window is displayed showing a list of all assets in the folders in which you have View permissions on Assets.



The screenshot shows the 'Link Assets from other folders' window. The title bar indicates the path: Unattended > Assets > Link Assets from other folders. The main area is titled '1 Select Asset' and contains a search bar with the placeholder 'Search'. Below the search bar, it says '0 rows selected'. A table lists five assets: 'Cred Asset 1', 'Cred Asset 2', 'Text Asset 1', and 'Text Asset 2'. Each asset has a checkbox, a 'Name' column, a 'Labels' column, and a 'Properties' column. The first two assets ('Cred Asset 1' and 'Cred Asset 2') have a blue circular icon with a white 'G' in the top-left corner, indicating they are linked to multiple folders. At the bottom of the table, there is a pagination bar showing '1 - 4 / 4' and 'Page 1 / 1'. To the right of the table, there are buttons for 'Cancel' and 'Continue'.

3. On the **Select Assets** section, select one or multiple assets from the list.
4. Click **Continue**. You are directed to the **Folder Validation** section. Here you can see the folders the assets are already linked to. If there are multiple folders, their names are displayed.

The screenshot shows a dark-themed dialog box titled 'Link Assets from other folders'. At the top left is a 'Select Asset' button with a pencil icon. At the top right is a 'Folder Validation' button with a blue circle containing a '2'. The main area contains a table with two rows:

Name	Folders
Cred Asset 1	@uipath.com's workspace, Important
Text Asset 1	Shared, @uipath.com's workspace, Important

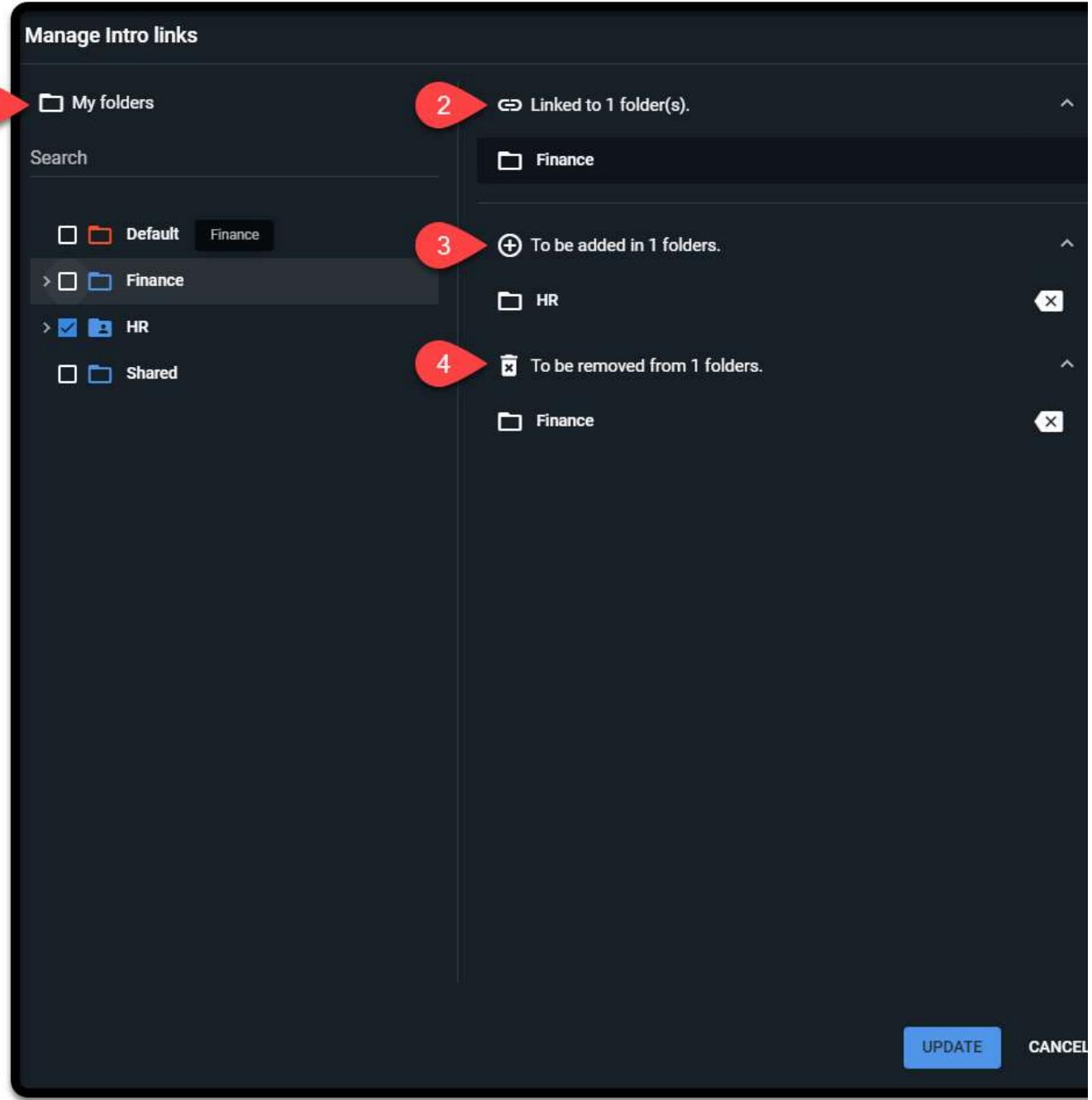
At the bottom right are 'Cancel' and 'Link' buttons. A horizontal scrollbar is visible at the bottom of the dialog.

5. Click **Remove** for the corresponding asset to revert the change or click **Exit** to cancel the operation.
6. Click **Link** if you want to make the link between the assets you selected and the current folder. The assets are displayed on the **Assets** page.

## Linking an Asset to Multiple Folders

1. Navigate to a folder the asset to be linked resides in.

Click **More Actions > Manage Links** for the desired asset to open the **Manage Links** window. The **Manage Asset Links** window is displayed.



1 - Left-hand pane displaying all the folders you have been granted **View** permissions on Assets.

2 - The current state of the asset displaying the number of folders it currently resides in as well as their names.

3 - The folders the asset is to be added in according to your selection in the left-hand pane.

4 - The folders the asset will be removed from.

1. Click **Update**. A confirmation window is displayed.

2. Click **Cancel** if you want to abort the changes or **Continue** for the changes to take effect. The operations are now reflected in Orchestrator according to your changes.

## Unlinking Assets From Folders

Unlinking assets from folders can be performed in a manner similar to the linking operation. Navigate to the link-management areas presented in the procedures above and remove the connections between a certain asset and a certain folder.

Alternatively, you can remove an asset using the **Remove** functionality. See [Removing an Asset](#).

**⚠️ IMPORTANT:**

Removing an asset that exists in multiple folders only removes it from the folder where the removal operation takes place, it does not remove it from the other folders as well. In order to completely delete an asset, you must remove all its existing links.

## Editing Assets

To edit an asset, click **More Actions > Edit** button, make the necessary changes and click **Update** in the **Update Asset** window. Only the asset value and name can be edited. All changes are audited and can be viewed in the [Audit](#) page. If you change the username of a credential asset, you have to also change the password.

## Adding Tags to Assets

**ⓘ NOTE:**

You need **Edit** on Assets and **View** on Tags to add existing tags to assets.

You need **Edit** on Assets and **Create** on Tags to add new tags to assets.

**ⓘ NOTE:**

- Each asset can have a maximum of one million key/value pairs.
- Labels and key/value properties are limited to 256 characters.
- Tag names can't contain these characters: <, >, %, &, ?, /, ::

You can apply tags to a asset either when creating one or editing an existing one. To add tags to a asset when editing it, follow these steps:

1. From the **Assets** page, click **More Actions > Edit** next to the desired asset. The asset is opened for editing.
2. On the **Labels** field, start typing the name of the label. You can choose an existing label or create a new one.
3. On the **Properties (key-value pairs)** field, click **Add new**.
4. Add new keys and values. You can choose existing keys and/or values or you can create new ones.
5. When done, click **Update**. Your asset is updated and the newly created tags, if any, become available for other objects.

## Removing Tags From Assets

To remove tags from a asset, follow these steps:

1. From the **Assets** page, click **More Actions > Edit** next to the desired asset. The asset is opened for editing.
2. On the **Labels** field, click the **X** adjacent to the name of the label to remove it. The label is removed.
3. On the **Properties (key-value pairs)** field, click the **X** adjacent to the keys and/or values to remove them. The keys and/or values are removed.
4. To delete a key/value pair click the **Remove** icon corresponding to that entry. The key/value pair is removed.
5. When done, click **Update**. Your asset is updated and tags are removed.

## Removing Assets

To remove an asset, click **More Actions > Remove**.

Alternatively, you can select one or multiple assets and click the **Remove** button.



Default Theme

English

# Studio User Guide

RELEASE: 2023.4

## TABLE OF CONTENTS

### [About Debugging](#)

Start Debugging

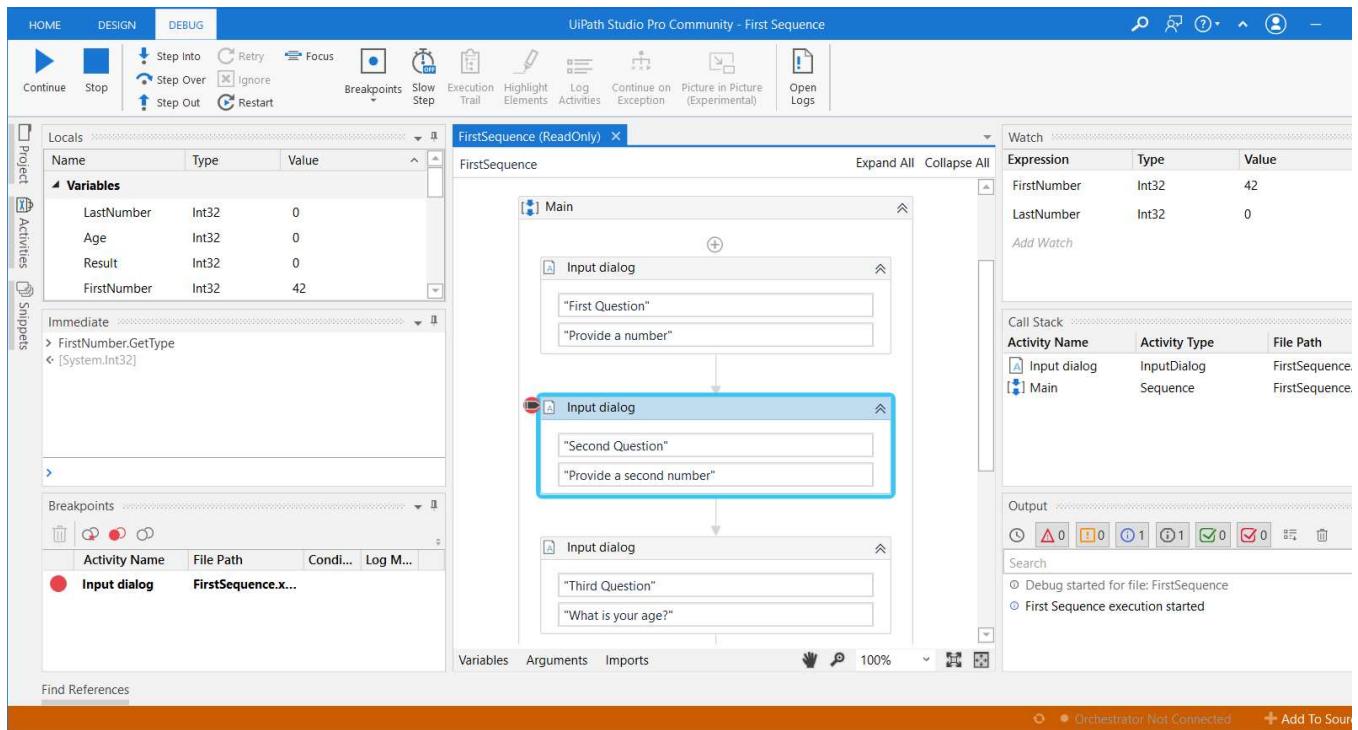
## About Debugging

Debugging is the process of identifying and removing errors that prevent the project from functioning correctly. It is recommended to perform debugging during the design stage of the automation project, at activity, file and project level.

Debugging can be performed using several options, defined in the ribbon and explained in the [Debugging Actions](#) page.

By default, debugging is performed on the local robot. You can run or debug your projects using a robot on a remote machine by enabling [remote debugging](#).

Several panels make it easier to view the debugging process, add values or monitor variables and arguments.



## Start Debugging

The options for running and debugging a file or project are available both in the **Design** and **Debug** tabs.

Option	Description
<b>Debug</b>	Click <b>Debug</b> or use <b>F5</b> to debug the whole project.
<b>Run</b>	Click <b>Run</b> or use <b>Ctrl + F5</b> to run the whole project.
<b>Debug File</b>	Click <b>Debug File</b> or use <b>F6</b> to debug the current file.
<b>Run File</b>	Click <b>Run File</b> or use <b>Ctrl + F6</b> to run the current file.

The default action under **Run/Debug** ribbon button can be configured from **Backstage View > Settings > Design tab**. Pick from **Debug File**, **Run File**, **Debug Project**, or **Run Project**, as the default action when clicking the button.

During debug, click the **Break** button to pause. The activity which is being debugged remains highlighted when paused. Once this happens, you can choose other debug actions like **Step Into** or press **Stop** to exit and return to design mode. The keyboard shortcut for the **Stop** button is **F12**.

It is recommended to use **Break** along with **Slow Step** so that you know exactly when debugging needs to be paused.

The **Continue** option is available when the debug process is paused.



Default Theme

English

# Six Debugging Panels in UiPath Studio



by [Parth Doshi](#) (/community-blog/parth-doshi/) • December 2, 2021



([https://twitter.com/intent/tweet?](https://twitter.com/intent/tweet?original_referer=https%3A%2F%2Fwww.uipath.com%2F&ref_src=twsrc%5Etfw&text=Six+Debugging+Panels+in+UiPath+Studio&tw_debugging_panels=true&url=https://www.uipath.com/blog/tutorials/6-in-debugging-panels)  
<https://www.linkedin.com/shareArticle?mini=true&url=https://www.uipath.com/blog/tutorials/6-in-debugging-panels>

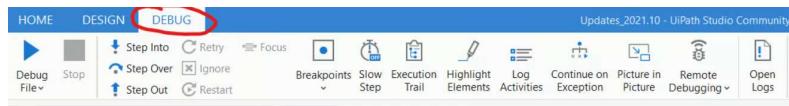
[https://www.facebook.com/sharer/sharer.php?](https://www.facebook.com/sharer/sharer.php?u=https://www.uipath.com/blog/tutorials/6-in-debugging-panels&t=Six+20Debugging%20Panels%20in%20UiPath%20Studio&summary=Six%20Debugging%20Panels%20in%20UiPath%20Studio&image=https://www.uipath.com/product/studio&title=Six%20Debugging%20Panels%20in%20UiPath%20Studio&url=https://www.uipath.com/product/studio)

In this article, it is significantly important to understand the debugging process in UiPath. The inaccuracies and errors pave the way for creating robust automation. UiPath Studio has some preeminent debugging features that can help you identify and acknowledge the errors that come your way when shaping your automation project.

In this article, you are going to discover the **six debugging panels** that you can find and explore in Studio which makes our development flow and problem solving easier. Before understanding each debugging panel in detail, let's quickly look at the debugging concept.

# What is debugging?

Debugging is a feature in Studio that allows a user to identify, understand, backtrack, and remove errors from the bots. This way you can make the bots more robust and functional as per the requirements and minimize the errors. The screenshot below shows where you can find the debugging option in Studio.



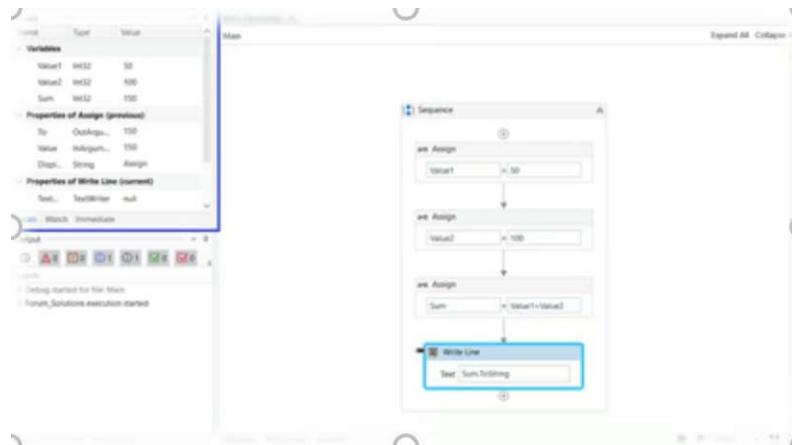
Now, let's dive into understanding each debugging panel.

## Debugging panels

### The locals panel

To easily understand what **Locals** panel is used for, I will relate the concept to a real-life example. Let's suppose a stranger comes to your local area and asks you for a place to visit. You will easily be able to give directions and guidance.

In the same way, Locals panel knows everything about a process such as variables, variable values, runtime values, and arguments, and you can easily view the values when you are running the workflow. Let us see a screenshot below of Locals panel.



### The watch panel

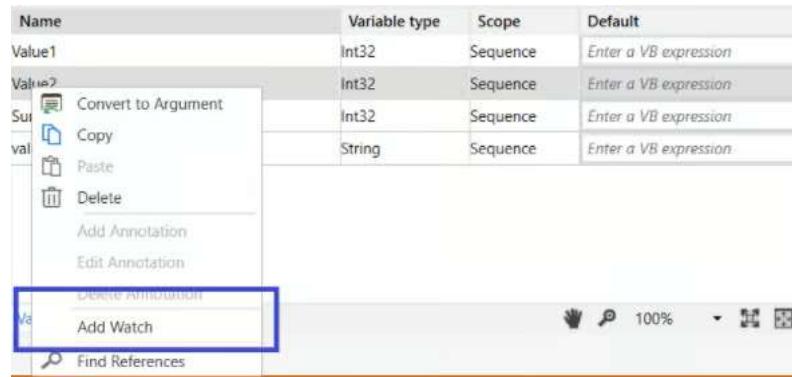
Watch Panel (as the name suggests) is used to watch something specific. In Studio, it is used to watch the variables and arguments values. And this is a particularly useful panel, while debugging, we do not have to search for the variables value in the output/local panel since we can directly monitor specific variables.

Now, let us understand diverse ways of adding variables to the Watch panel.

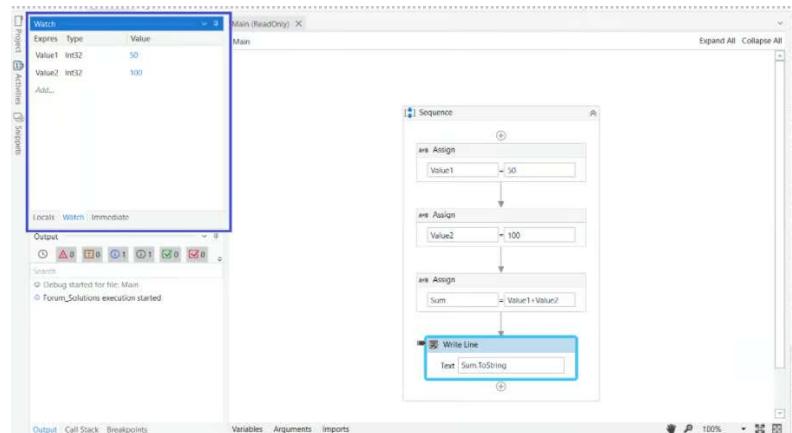
- 1) Add via right-click from Locals

Locals		
Name	Type	Value
<b>Variables</b>		
Value1	Int32	0
Value	Add to Watch	0
Sum	Int32	0
value	String	null
<b>Properties of Sequence (previous)</b>		
DisplayName	String	Sequence
<b>Properties of Assign: Value 1 (current)</b>		
➤ To	OutArgument	OutArgument<int>   ArgumentType...
➤ Value	InArgument	50

## 2) Add via the add-in Watch panel



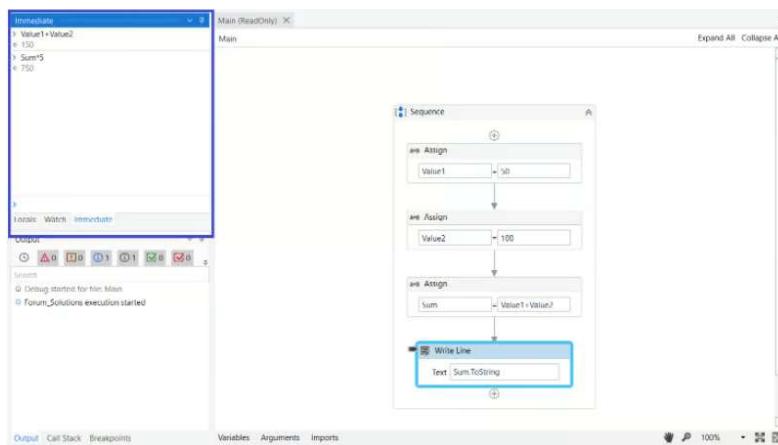
See the screenshot below of Watch panel.



## The immediate panel

Immediate panel in debugging is used to act immediately. The Immediate panel goal is to see values, variables, or arguments and to evaluate an expression and output the result. For example, if you have two integer values A & B and you want to see the value of  $A*B$ , you can do that in the Immediate panel.

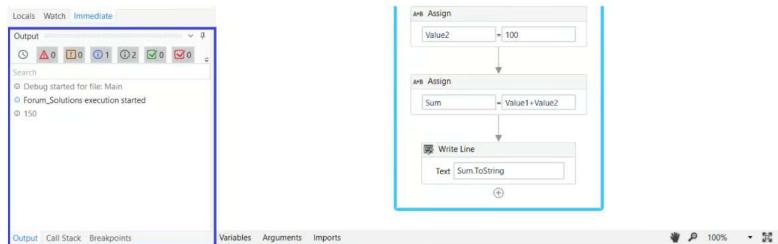
Let us see a similar example in the screenshot below.



## The output panel

Output panel is used to see the log message or any custom value that we are writing using **Log Message** or **Write Line activity**. It is also used to notify information regarding packages installation, remote debugging states, etc.

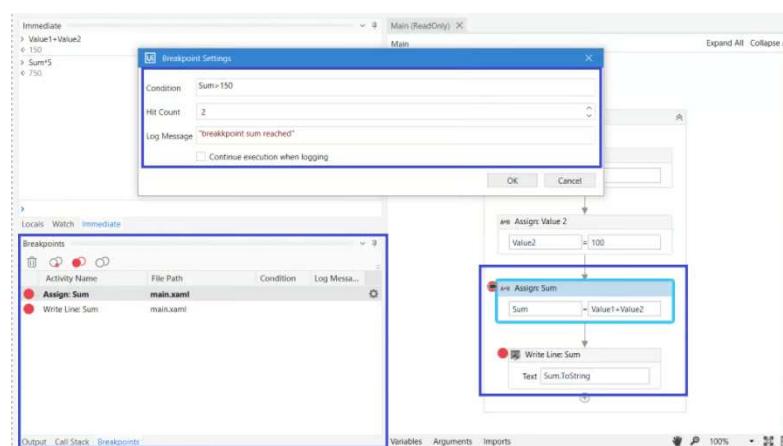
**Note:** The output panel is also available outside the debugging process. It also shows the execution start and end time in the screenshot below. This panel also categorizes the log messages into distinct categories like warning, information, trace, error, and test results in Failed and Passed Assertions.



## The breakpoints panel

Breakpoints are used to pause workflow at any given point and monitor the values or check if the workflow is running as expected. Breakpoints panel helps to monitor the breakpoints in the workflow. It shows which activity has the breakpoint and is present in which **Extensible Application Markup Language**. It is also used to give conditions at break points.

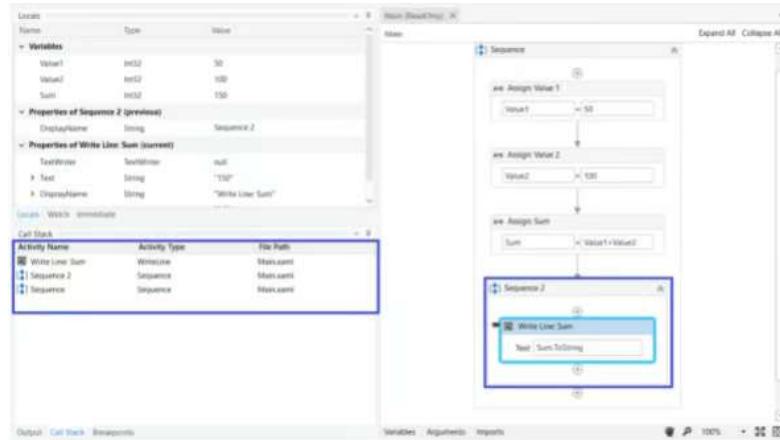
The screenshot below explains every aspect of Breakpoints and Breakpoints Panel.



# The call stack panel

The Call Stack panel is used to see the upcoming task to be performed within a particular sequence. It shows three things: **activity name**, **activity type**, **file path**.

The Call Stack panel is immensely helpful when there are heavy workflows. We can use it to highlight the currently executing activity within the workflow. Below is a screenshot of the Call Stack panel.



## Conclusion

Debugging is an important process to identify and remove errors. Generally, it is recommended to perform it during the design stage of the automation project to assure high quality. As we discovered there are several panels that makes it easier to view the debugging process, add values or monitor variables and arguments.

---

Topics: [Studio\(/community-blog/studio/\)](#)

---



[Parth Doshi \(/community-blog/parth-doshi/\)](#)

Consultant, WonderBotz

## Related articles



TUTORIALS

February 8, 2023

## How to Use the Try-Catch Activity in UiPath Studio (/community-blog/tutorials/how-to-use-the-try-catch-activity-in-uipath-studio).

(/community-blog/tutorials/how-to-use-the-try-catch-activity-in-uipath-studio)

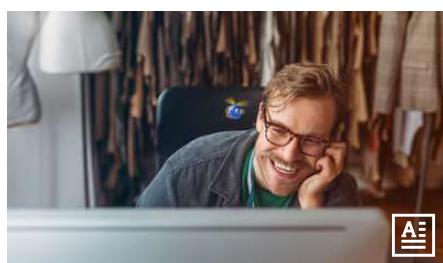


TUTORIALS

January 19, 2023

## How RPA Developers Can Kick Start SAP Automation (/community-blog/tutorials/how-rpa-developers-can-kick-start-sap-automation).

(/community-blog/tutorials/how-rpa-developers-can-kick-start-sap-automation)



TUTORIALS

January 9, 2023

## How to Create Test Automation with UiPath Studio (/community-blog/tutorials/how-to-create-test-automation-with-uipath-studio).

(/community-blog/tutorials/how-to-create-test-automation-with-uipath-studio)

PLATFORM	AUTOMATE	SOLUTIONS & RESOURCES	PARTNERS	COMPANY
<a href="https://www.uipath.com/product">Explore the Platform</a>	Studio Family	SOLUTIONS By industry	RESOURCE CENTER	<a href="#">UiPath Partner Network</a>
<a href="https://www.uipath.com/pricing">Plans and Pricing</a>	Apps	By process	( <a href="#">https://www.uipath.com/partner</a> )	<a href="#">About us</a>
<a href="https://www.uipath.com/support">Customer support</a>	code-app-studio), Integration Service	Technology solutions overview	( <a href="#">https://www.uipath.com/partner</a> )	<a href="#">Network partners</a>
DISCOVER	api-integration-	( <a href="#">https://www.uipath.com/partner</a> )	( <a href="#">https://www.uipath.com/partner</a> )	<a href="#">Careers</a>
Process Mining	automation)	LEARNING	case-studies)	( <a href="#">https://www.uipath.com/partner</a> )
( <a href="#">https://www.uipath.com/product/process-mining</a> ).	Assistant	Academy	White papers	<a href="#">Newsroom</a>
Task Mining	robot-assistant).	Academic Alliance	whitepapers).	<a href="#">Grow as a partner</a>
( <a href="#">https://www.uipath.com/product/task-mining</a> ).	Robots	( <a href="#">https://www.uipath.com/partner</a> )	Portal	<a href="#">Investor Relations</a>
Communications Mining	Action Center™	Certifications	( <a href="#">https://www.uipath.com/partner</a> )	( <a href="#">https://partnerportal.uipath.com/</a> )
( <a href="#">https://www.uipath.com/product/communications-mining</a> ).	( <a href="#">https://www.uipath.com/partner</a> )	What is RPA	( <a href="#">https://www.uipath.com/partner</a> )	<a href="#">Trust &amp; security</a>
Automation Hub	Data Service	( <a href="#">https://www.uipath.com/partner</a> )	Demos and videos	( <a href="#">https://www.uipath.com/partner</a> )
( <a href="#">https://www.uipath.com/product/automation-code-data-modeling</a> ).	( <a href="#">https://www.uipath.com/partner</a> )	SUPPORT & COMMUNITY	( <a href="#">https://www.uipath.com/partner</a> )	<a href="#">UiPath Gift Shop</a>
Marketplace	Document Understanding	Customer support	EVENTS	( <a href="#">https://bamkostores.cc</a> )
( <a href="#">https://marketplace.uipath.com/</a> ).	( <a href="#">https://www.uipath.com/partner</a> )	( <a href="#">https://www.uipath.com/partner</a> ) FORWARD	( <a href="#">https://www.uipath.com/partner</a> )	<a href="#">UiPath Foundation</a>
AI Center™	Customer Portal	( <a href="#">https://www.uipath.com/partner</a> )	( <a href="#">https://www.uipath.com/partner</a> )	( <a href="#">https://uipathfoundation.com/</a> )
( <a href="#">https://www.uipath.com/partner</a> )	ai-integration-with-ai-center)	Documentation	( <a href="#">https://www.uipath.com/partner</a> )	<a href="#">Code of Conduct</a>
OPERATE	FORUM	Conferences	( <a href="#">https://www.uipath.com/partner</a> )	( <a href="#">https://www.uipath.com/partner</a> )
Insights	( <a href="#">https://www.uipath.com/partner</a> )	( <a href="#">https://www.uipath.com/partner</a> )	( <a href="#">https://www.uipath.com/partner</a> )	<a href="#">Report Ethical Concerns</a>
( <a href="#">https://www.uipath.com/product/test-insights</a> )	Blog	UiPath DevCon	( <a href="#">https://www.uipath.com/partner</a> )	( <a href="#">https://secure.ethicspoint.eu/</a> )
Test Manager	( <a href="#">https://www.uipath.com/partner</a> )	( <a href="#">https://www.uipath.com/partner</a> )	( <a href="#">https://www.uipath.com/partner</a> )	<a href="#">Employment Scams</a>
( <a href="#">https://www.uipath.com/partner</a> )	events manager)	( <a href="#">https://www.uipath.com/partner</a> )	( <a href="#">https://www.uipath.com/partner</a> )	( <a href="#">https://www.uipath.com/partner</a> )
Orchestrator	Service status	See all events	( <a href="#">https://www.uipath.com/partner</a> )	<a href="#">job-scams</a> )
( <a href="#">https://www.uipath.com/partner</a> )	( <a href="#">https://www.uipath.com/partner</a> )	( <a href="#">https://www.uipath.com/partner</a> )	( <a href="#">https://www.uipath.com/partner</a> )	
Automation Ops	( <a href="#">https://www.uipath.com/partner</a> )	( <a href="#">https://www.uipath.com/partner</a> )	( <a href="#">https://www.uipath.com/partner</a> )	
( <a href="#">https://www.uipath.com/partner</a> )	ops)	( <a href="#">https://www.uipath.com/partner</a> )	( <a href="#">https://www.uipath.com/partner</a> )	
Automation Cloud™	( <a href="#">https://www.uipath.com/partner</a> )	( <a href="#">https://www.uipath.com/partner</a> )	( <a href="#">https://www.uipath.com/partner</a> )	
( <a href="#">https://www.uipath.com/partner</a> )	cloud)	( <a href="#">https://www.uipath.com/partner</a> )	( <a href="#">https://www.uipath.com/partner</a> )	

**Automation Suite**

(<https://www.uipath.com/product/automation-suite>)

---

Let's connect



(<https://facebook.com/uipath>)



(<https://www.linkedin.com/company/uipath>)



(<https://twitter.com/uipath>)

---

[Trust & security](#) (<https://www.uipath.com/legal/trust-and-security>) • [Terms of Use](#) (<https://www.uipath.com/legal/terms-of-use>) •

[Privacy Policy](#) (<https://www.uipath.com/legal/privacy-policy>) • [Cookies Policy](#) (<https://www.uipath.com/legal/cookies-policy>) • [Ver configuración de cookies](#)

© 2005-2023 UiPath. All rights reserved.

# Studio User Guide

RELEASE: 2023.4 ▾

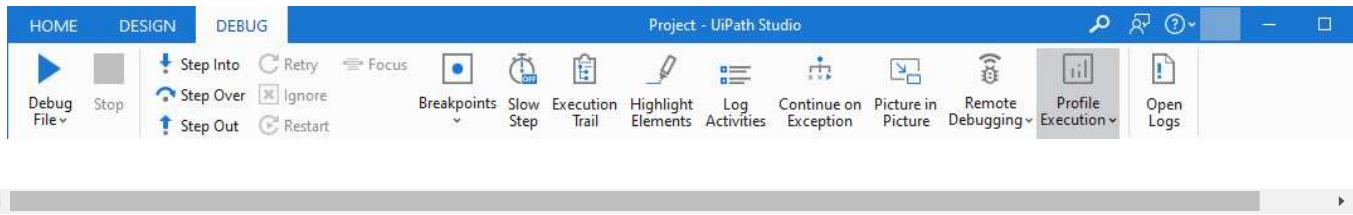
## TABLE OF CONTENTS

### [Debugging Actions](#)

- [Step Into](#)
- [Step Over](#)
- [Step Out](#)
- [Retry](#)
- [Ignore](#)
- [Restart](#)
- [Break](#)
- [Focus](#)
- [Slow Step](#)
- [Execution Trail](#)
- [Highlight Elements](#)
- [Log Activities](#)
- [Continue on Exception](#)
- [Picture in Picture](#)
- [Remote Debugging](#)
- [Profile Execution](#)
- [Open Logs](#)

## Debugging Actions

Debugging of a single file or the whole project can be performed both from the **Design** or **Debug** ribbon tabs. However, the debugging process is not available if project files have validation errors.



## Step Into

Use **Step Into** to debug activities one at a time. When this action is triggered, the debugger opens and highlights the activity before it is executed.

When **Step Into** is used with **Invoke Workflow File** activities, the workflow is opened in a new tab in **ReadOnly** mode and each activity is executed one by one.

The keyboard shortcut for **Step Into** is **F11**.

## Step Over

Unlike the **Step Into** action, **Step Over** does not open the current container. When used, the action debugs the next activity, highlighting containers (such as flowcharts, sequences, or **Invoke Workflow File** activities) without opening them.

This action comes in handy for skipping analysis of large containers which are unlikely to trigger any issues during execution.

**Step Over** is available using the **F10** keyboard shortcut.

## Step Out

As the name suggests, this action is used for stepping out and pausing the execution at the level of the current container. **Step Out** completes the execution of activities in the current container, before pausing the debugging. This option works well with nested sequences.

**Step Out** is available using the **Shift + F11** keyboard shortcut.

## Retry

**Retry** re-executes the previous activity, and throws the exception if it's encountered again. The activity which threw the exception is highlighted and details about the error are shown in the **Locals** and **Call Stack** panels.

## Ignore

The **Ignore** action can be used to ignore an encountered exception and continue the execution from the next activity so that the rest of the workflow can be debugged.

This action is useful when jumping over the activity that threw the exception and continuing debugging the remaining part of the project.

## Restart

**Restart** is available after an exception was thrown and the debug process is paused. The action is used for restarting the debugging process from the first activity of the project. Use **Slow Step** to slow down the debugging speed and properly inspect activities as they are executed.

Please take into consideration that when using this option after using the **Run from this Activity** action, the debugging is restarted from the previously indicated activity.

## Break

**Break** allows you to pause the debugging process at any given moment. The activity which is being debugged remains highlighted when paused. Once this happens, you can choose to **Continue**, **Step Into**, **Step Over**, or **Stop** the debugging process.

It is recommended to use **Break** along with **Slow Step** so that you know exactly when debugging needs to be paused.

An alternative to using **Slow Step** in this situation is to keep an eye on the **Output** panel and use **Break** on the activity that is currently being debugged.

## Focus

**Focus Execution Point** helps you return to the current breakpoint or the activity that caused an error during debugging. The **Focus** button is used after navigating through the process, as an easy way to return to the activity that caused the error and resume the debugging process.

Alternatively, when debugging is paused because a breakpoint was reached, **Focus** can be used for returning to said breakpoint, after navigating through activities contained in the automation process.

A third case is when the debugging is paused either after using **Step Into** or **Step Over** and then navigating through the process. In this case, **Focus** returns to the activity that paused the debugging process.

From the **Breakpoints** context menu, you can select **Focus** to highlight the activity with the breakpoint.

## Slow Step

**Slow Step** enables you to take a closer look at any activity during debugging. While this action is enabled, activities are highlighted in the debugging process. Moreover, containers such as flowcharts, sequences, or **Invoke Workflow File** activities are opened. This is similar to using **Step Into**, but without having to pause the debugging process.

**Slow Step** can be activated both before or during the debugging process. Activating the action does not pause debugging.

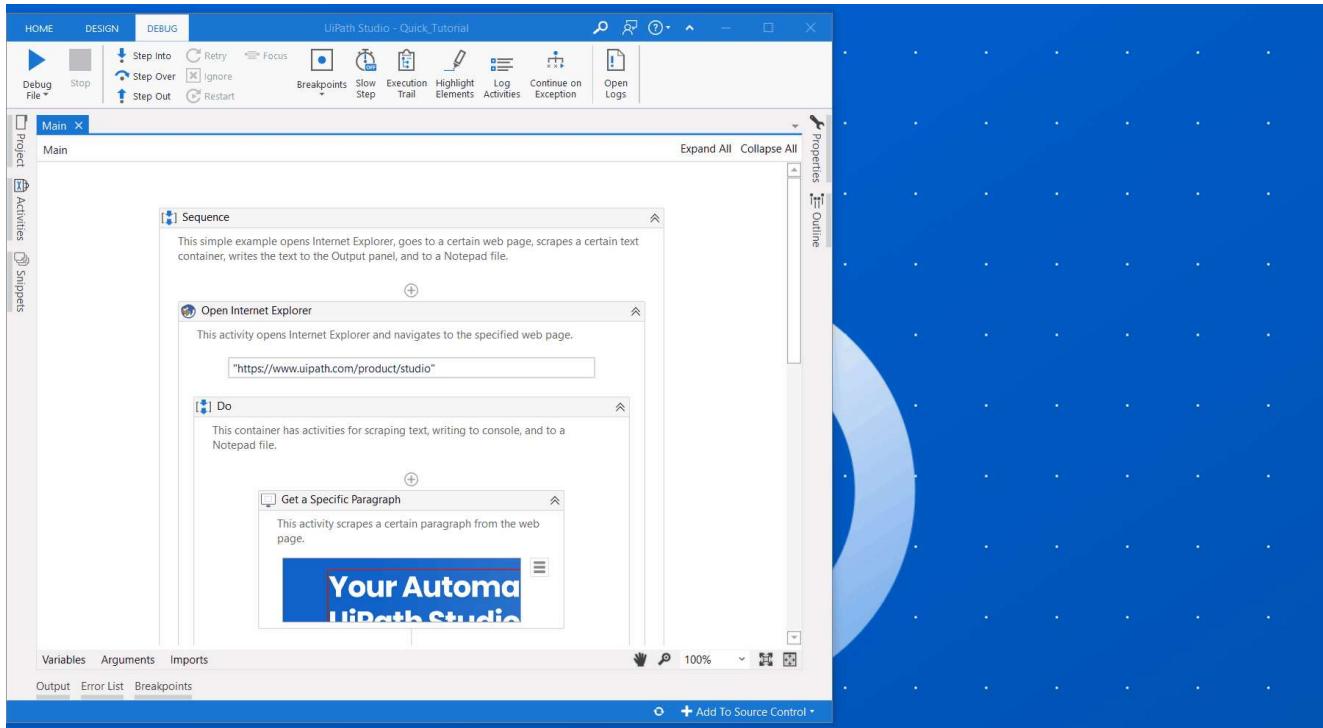
Although called **Slow Step**, the action comes with 4 different speeds. The selected speed step runs the debugging process slower than the previous one. For example, debugging with **Slow Step** at 1x runs it the slowest, and fastest at 4x. In other words, the speed dictates how fast the debugger jumps from one activity to the next.

Each time you click **Slow Step** the speed changes by one step. You can easily tell by the icon, which updates accordingly.

## Execution Trail

The **Execution Trail** ribbon button is disabled by default. When enabled, it shows the exact execution path at debugging. As the process is executed, each activity is highlighted and marked in the **Designer** panel, showing you the execution as it happens:

- executed activities are marked and highlighted in green;
- activities that were not executed are not marked in any way;
- activities that threw an exception are marked and highlighted in red.



## Highlight Elements

If enabled, UI elements are highlighted during debugging. The option can be used both with regular and step-by-step debugging.

## Log Activities

If enabled, debugged activities are displayed as Trace logs in the **Output** panel. Note that **Highlight Elements** and **Log Activities** options can only be toggled before debugging, and persist when reopening the automation project. This is not applicable for invoked workflows unless these files are opened in the **Designer** panel.

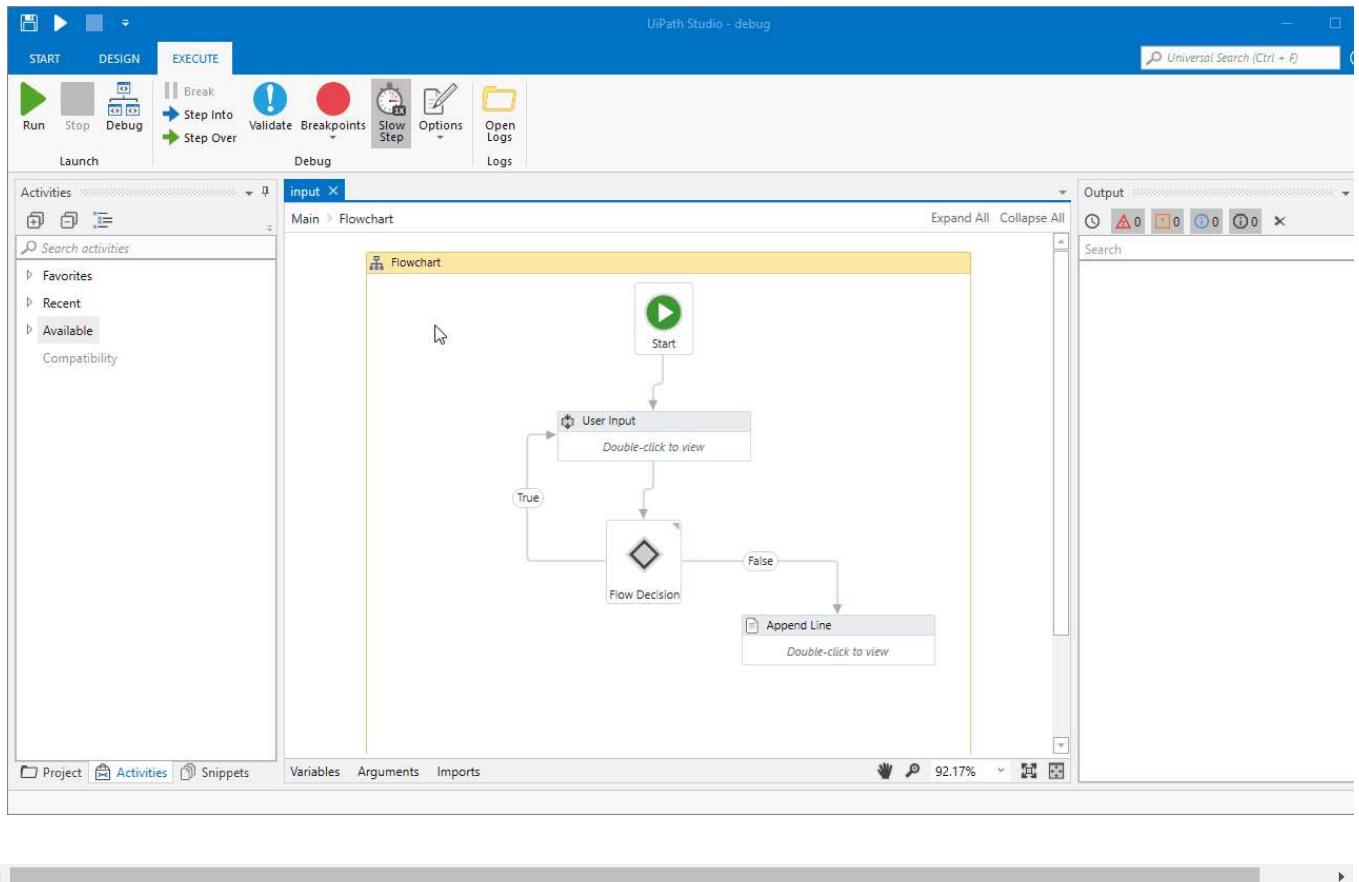
Logs are automatically sent to Orchestrator if connected, but you can have them stored locally by disabling the **Allow Development Logging** option from the **Robot Settings** tab in the Add or Edit user window.

Disabling **Log Activities** can be a way to send smaller log files to Orchestrator.

### **NOTE:**

When running processes from Studio, the logs sent to Orchestrator are always Trace when Log Activities is disabled, and Verbose when Log Activities is enabled. This overrides both the Robot and the Orchestrator setting.

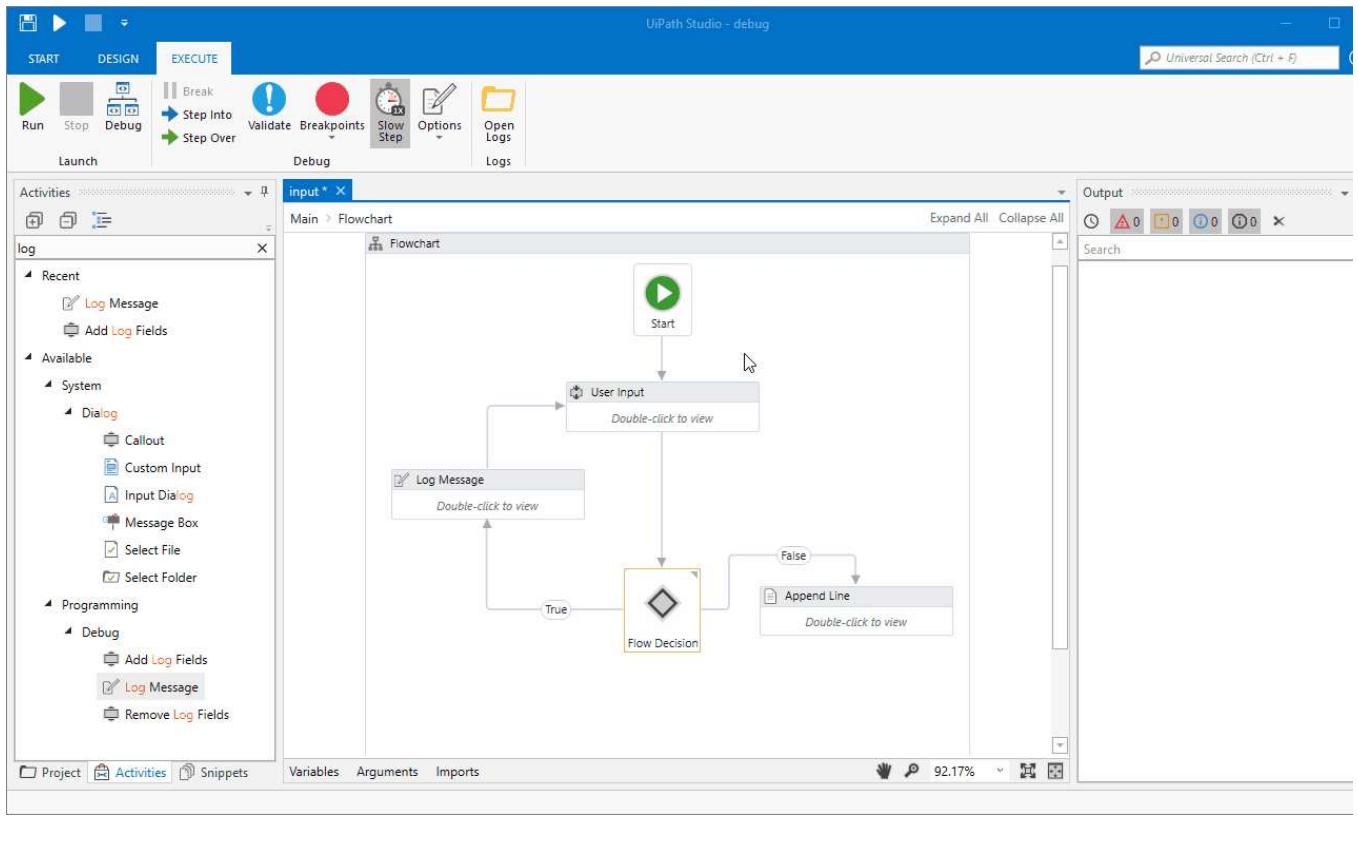
By default, the debugger logs activities so that each step appears in the **Output** panel. We recommend leaving it enabled for easier tracing, as you can see in the image below:



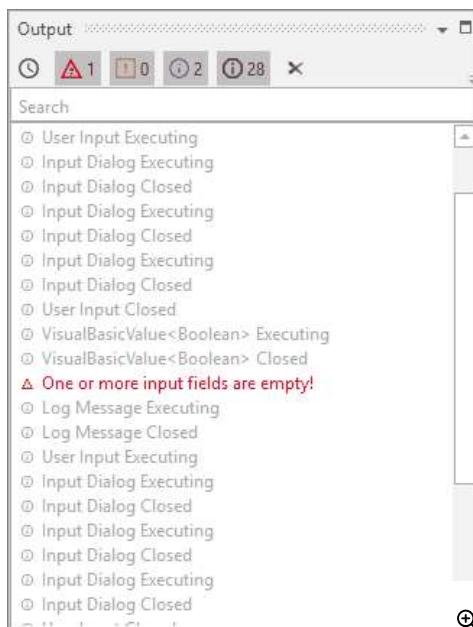
The issue here is that one or more input fields from the User Input sequence are blank, which is a True condition for the Flow Decision. You can tell this by the fact that, during debugging, the User Input sequence is executed twice, meaning that one or more fields were left blank during the first execution.

If you decide to disable the **Log Activities** option for debugging, Trace logs are not displayed in the **Output** panel. In the case of a normal execution with no errors, you only get to see the debug execution start and end times. However, adding a **Log Message** can help you determine where issues might occur.

For example, you can add a **Log Message** activity to inform you that, in this case, one or more input fields are empty. This message appears in the **Output** panel during debugging, even if the **Log Activities** option is disabled, as you can see below:



Remember that you can always filter the messages displayed in the [Output panel](#) by simply selecting the alert types of interest, or even clear all messages.



Note that by default all debugging logs are sent to Orchestrator. You can disable this by clearing the **Allow Development Logging** option from the [Settings tab](#) in the Add or Edit Robot window. If this option is disabled, debugging logs are only stored locally.

## Continue on Exception

This debugging feature is disabled by default. When disabled in the ribbon, it throws the execution error and stops the debugging, highlights the activity which threw the exception, and logs the exception in the [Output panel](#). If a [Global Exception Handler](#) was previously set in the

project, the exception is passed on to the handler.

When enabled, the exception is logged in the **Output** panel, the execution continues.

## Picture in Picture

The **Picture in Picture** ribbon option in the **Debug** tab is available for both executing and debugging processes or libraries in a separate session on your machine.

If enabled, whenever you select **Run** or **Run File**, **Debug** or **Debug File** the process starts either in a separate session or in a virtual desktop in the user session. If **Picture in Picture** is disabled, debugging and execution is performed in the current session.

Having the option to run a process in Picture in Picture (PiP) can be very useful in attended automation. Verify whether a process runs successfully in PiP, and then update the project settings to indicate if it can be executed using this feature after it is published:

1. In the **Project** panel, click **Settings**  to open the Project Settings window.

2. In the **General** tab:

- **PiP Options** - Indicate whether the project was tested using Picture in Picture and whether it should start in PiP by default.
  - **Tested for PiP usage; Starting in PiP** - The automation has been approved to run in PiP mode. When run, it starts in PiP by default.
  - **Tested for PiP usage; Not starting in PiP by default** - The automation has been approved to run in PiP mode. When run, it starts in the main session or desktop by default.
  - **Not tested for PiP usage** - The automation has not been approved to run in PiP mode. When run, it starts in the main session or desktop by default. If run in PiP, a dialog informs the user it was not tested using this feature and prompts for confirmation before proceeding.
- **PiP Type** - Select how to isolate the automation from the user session when running the project in PiP: **New Session** (child session on the machine) or **New Desktop** (virtual desktop in the user session).

For more information, including limitations of this feature, see [Picture in Picture](#) in the Robot Guide.

## Remote Debugging

When this feature is enabled, all run and debug operations are performed on a specified remote robot instead of the robot installed locally, allowing you to test the automation on different environments. For more information, see [Remote Debugging](#).

## Profile Execution

You can identify performance bottlenecks in the workflow when you debug the file. For more information, see [Profile Execution](#).

## Open Logs

Clicking **Open Logs** brings up the %localappdata%\UiPath\Logs folder where logs are locally stored. The naming format of log files is YYYY-DD-MM\_Component.log (such as 2018-09-12\_Execution.log, or 2018-09-12\_Studio.log). Read more about logging [here](#).



Default Theme

English

# Studio User Guide

RELEASE:

2023.4



## TABLE OF CONTENTS

### [The Call Stack Panel](#)

## The Call Stack Panel

The **Call Stack** panel displays the next activity to be executed and its parent containers when the project is paused in debugging.

The panel is displayed during execution in debug mode and it gets populated after using **Step Into**, **Break**, **Slow Step**, or after the execution was paused because an error or a breakpoint was encountered.

Activity Name	File Path
Message Box	Seq.xaml
Sequence	Seq.xaml
Invoke Workflow File	Main.xaml
Sequence	Main.xaml

Double-clicking an item in the **Call Stack** panel, focuses and highlights the selected activity in the **Designer** panel.

If during debugging, an activity throws an exception, it is marked in the **Call Stack** panel and the activity is highlighted in red.



Default Theme

English

# Studio User Guide

RELEASE: 2023.4

## TABLE OF CONTENTS

### [Test Activities](#)

[Test Activity](#)[Create Test Bench](#)[Run to this Activity](#)[Run from this Activity](#)

## Test Activities

### Test Activity

The **Test Activity** context menu option—part of the **Designer** panel—is used for running a test on the currently selected activity. When clicked, the **Locals** panel opens displaying the variables and arguments in scope.

Test Activity can be used in two ways:

- Add default values to properties and test.
- Add arguments and/or properties to activity properties and use the **Local** panel to add values after clicking the **Test Activity** option.

Double-click the value field of a variable or argument, or click the  icon in the **Locals** panel, and add a new value. Next, click **Step Into** to focus and execute the activity, and monitor the variable or argument's value in the **Locals** panel.

The same is available when clicking **Continue**, but the values are not visible in the **Locals** panel.

Please take into consideration that dynamic checks when variables depend on other variables that are defined later are not supported.

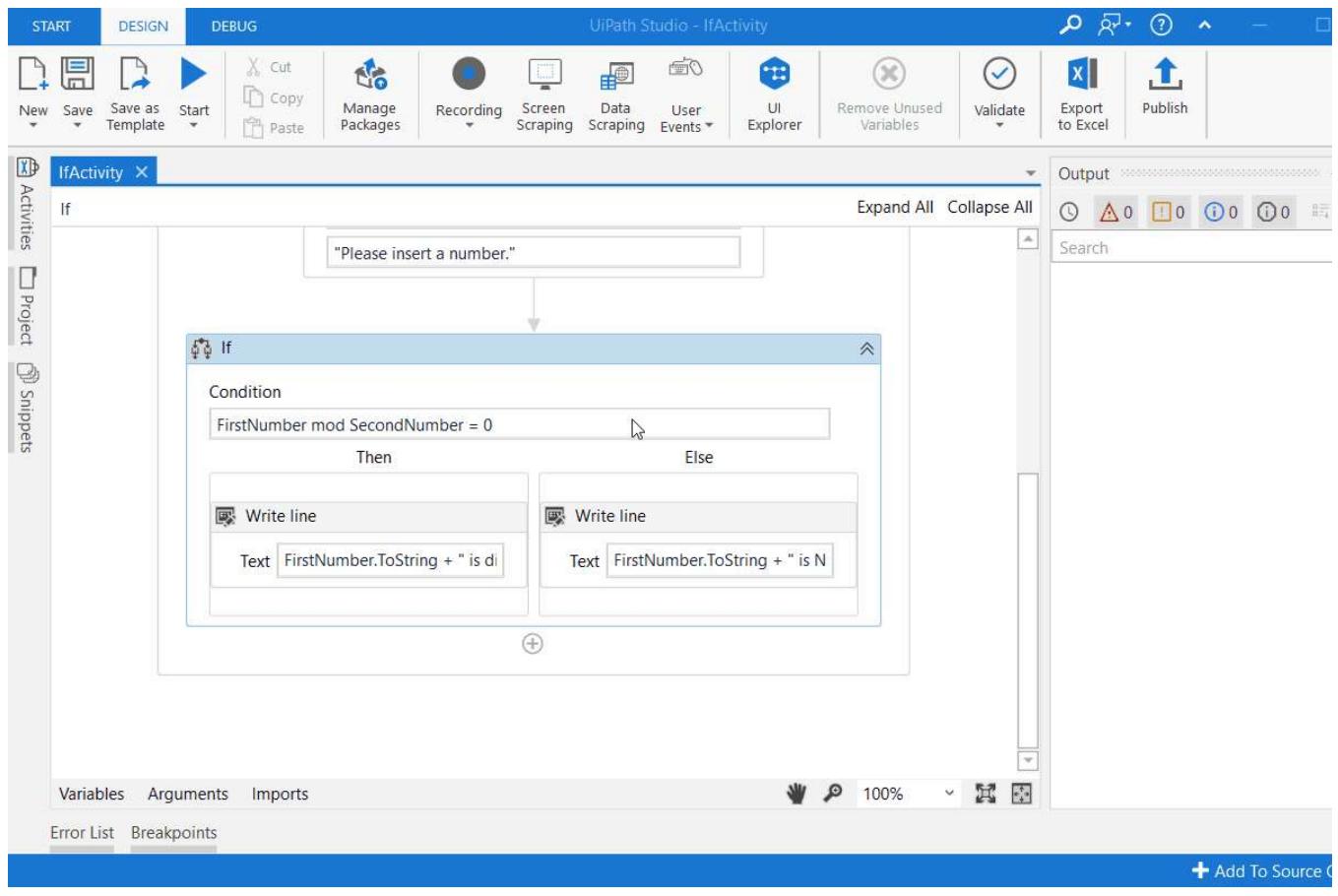
Execution logs generated by the **Test Activity** action are visible in the **Output** panel. Exceptions in Studio can be bubbled up, which means that the exception may be passed to parent containers in case it may be handled by them.

The **Test Activity** option is not available during debugging.

#### Example of Using Test Activity

For the [If Activity](#) example, we created a process that asks the user for two numbers, checks to see if one is divisible by the other, and depending on the result, displays a different message in the **Output** panel.

To check the behavior of the **If** activity defined in the process, use the **Test Activity** option, as illustrated below:



The **Test Activity** action places the activity in the debugger and asks you for values to variables. Once provided, click the **Continue** button for the debugging process to continue. In this particular case, a message was written in the **Output** panel with the correct answer, which means that the expressions written in the **If** activity were correct.

## Create Test Bench

The **Create Test Bench** option allows for the creation of automation building blocks, which can then be tested and added to the final workflow.

It is used for testing activities, working with variables and debugging the process. All this is done in a test bench workflow, a temporary sequence that's not part of the current project and that is discarded when closed.

The **Create Test Bench** option is similar to the **Test Activity** option, with the exception that the latter is contained and defined in an actual workflow.

To use the **Create Test Bench** option, go to the **Activities** panel search bar or use **Ctrl + Alt + F** keyboard shortcut. Type the name of an activity and right-click to open the context menu.

Select **Create Test Bench** and the activity is automatically added to a sequence file not included in your project. From there you can add other activities, change their default properties and debug the process. The **Output** panel displays any logs or errors found during the debugging.

To save the file to your project, simply use the ribbon option **Save as**, add a file name and save it to the same file path as your project.

Please note that **Create Test Bench** does not work with **Pick Branch** activity.

## Run to this Activity

The **Run to this Activity** option is available when right-clicking an activity in the **Designer** panel.

This option starts the debugging process and pauses before the selected activity is executed while highlighting it in the panel. If **Run to this Activity** is triggered when debugging is already paused, the execution continues until the activity is reached.

## Run from this Activity

The **Run from this Activity** context menu option enters debugging in a pause state, allowing you to make changes to the values of variables and arguments from the **Locals** panel. Press **Continue** to start debugging or use actions such as **Step Into**, **Step Over**, **Step Out**.

 **NOTE:**

- When used for an activity inside a scope, the scope activity is also executed.
- An error occurs if you use **Run from this Activity** for an activity added inside one of the following scope activities: **Try Catch**, **Switch**, **Parallel**, **Pick**, **Trigger Scope**, or **Retry Scope**.



Default Theme

English

# Studio User Guide

RELEASE: 2023.4 ▾

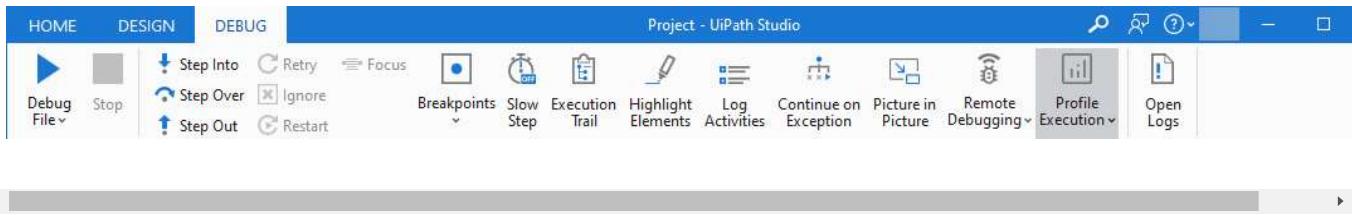
## TABLE OF CONTENTS

### [Debugging Actions](#)

- [Step Into](#)
- [Step Over](#)
- [Step Out](#)
- [Retry](#)
- [Ignore](#)
- [Restart](#)
- [Break](#)
- [Focus](#)
- [Slow Step](#)
- [Execution Trail](#)
- [Highlight Elements](#)
- [Log Activities](#)
- [Continue on Exception](#)
- [Picture in Picture](#)
- [Remote Debugging](#)
- [Profile Execution](#)
- [Open Logs](#)

## Debugging Actions

Debugging of a single file or the whole project can be performed both from the **Design** or **Debug** ribbon tabs. However, the debugging process is not available if project files have validation errors.



## Step Into

Use **Step Into** to debug activities one at a time. When this action is triggered, the debugger opens and highlights the activity before it is executed.

When **Step Into** is used with **Invoke Workflow File** activities, the workflow is opened in a new tab in **ReadOnly** mode and each activity is executed one by one.

The keyboard shortcut for **Step Into** is **F11**.

## Step Over

Unlike the **Step Into** action, **Step Over** does not open the current container. When used, the action debugs the next activity, highlighting containers (such as flowcharts, sequences, or **Invoke Workflow File** activities) without opening them.

This action comes in handy for skipping analysis of large containers which are unlikely to trigger any issues during execution.

**Step Over** is available using the **F10** keyboard shortcut.

## Step Out

As the name suggests, this action is used for stepping out and pausing the execution at the level of the current container. **Step Out** completes the execution of activities in the current container, before pausing the debugging. This option works well with nested sequences.

**Step Out** is available using the **Shift + F11** keyboard shortcut.

## Retry

**Retry** re-executes the previous activity, and throws the exception if it's encountered again. The activity which threw the exception is highlighted and details about the error are shown in the **Locals** and **Call Stack** panels.

## Ignore

The **Ignore** action can be used to ignore an encountered exception and continue the execution from the next activity so that the rest of the workflow can be debugged.

This action is useful when jumping over the activity that threw the exception and continuing debugging the remaining part of the project.

## Restart

**Restart** is available after an exception was thrown and the debug process is paused. The action is used for restarting the debugging process from the first activity of the project. Use **Slow Step** to slow down the debugging speed and properly inspect activities as they are executed.

Please take into consideration that when using this option after using the **Run from this Activity** action, the debugging is restarted from the previously indicated activity.

## Break

**Break** allows you to pause the debugging process at any given moment. The activity which is being debugged remains highlighted when paused. Once this happens, you can choose to **Continue**, **Step Into**, **Step Over**, or **Stop** the debugging process.

It is recommended to use **Break** along with **Slow Step** so that you know exactly when debugging needs to be paused.

An alternative to using **Slow Step** in this situation is to keep an eye on the **Output** panel and use **Break** on the activity that is currently being debugged.

## Focus

**Focus Execution Point** helps you return to the current breakpoint or the activity that caused an error during debugging. The **Focus** button is used after navigating through the process, as an easy way to return to the activity that caused the error and resume the debugging process.

Alternatively, when debugging is paused because a breakpoint was reached, **Focus** can be used for returning to said breakpoint, after navigating through activities contained in the automation process.

A third case is when the debugging is paused either after using **Step Into** or **Step Over** and then navigating through the process. In this case, **Focus** returns to the activity that paused the debugging process.

From the **Breakpoints** context menu, you can select **Focus** to highlight the activity with the breakpoint.

## Slow Step

**Slow Step** enables you to take a closer look at any activity during debugging. While this action is enabled, activities are highlighted in the debugging process. Moreover, containers such as flowcharts, sequences, or **Invoke Workflow File** activities are opened. This is similar to using **Step Into**, but without having to pause the debugging process.

**Slow Step** can be activated both before or during the debugging process. Activating the action does not pause debugging.

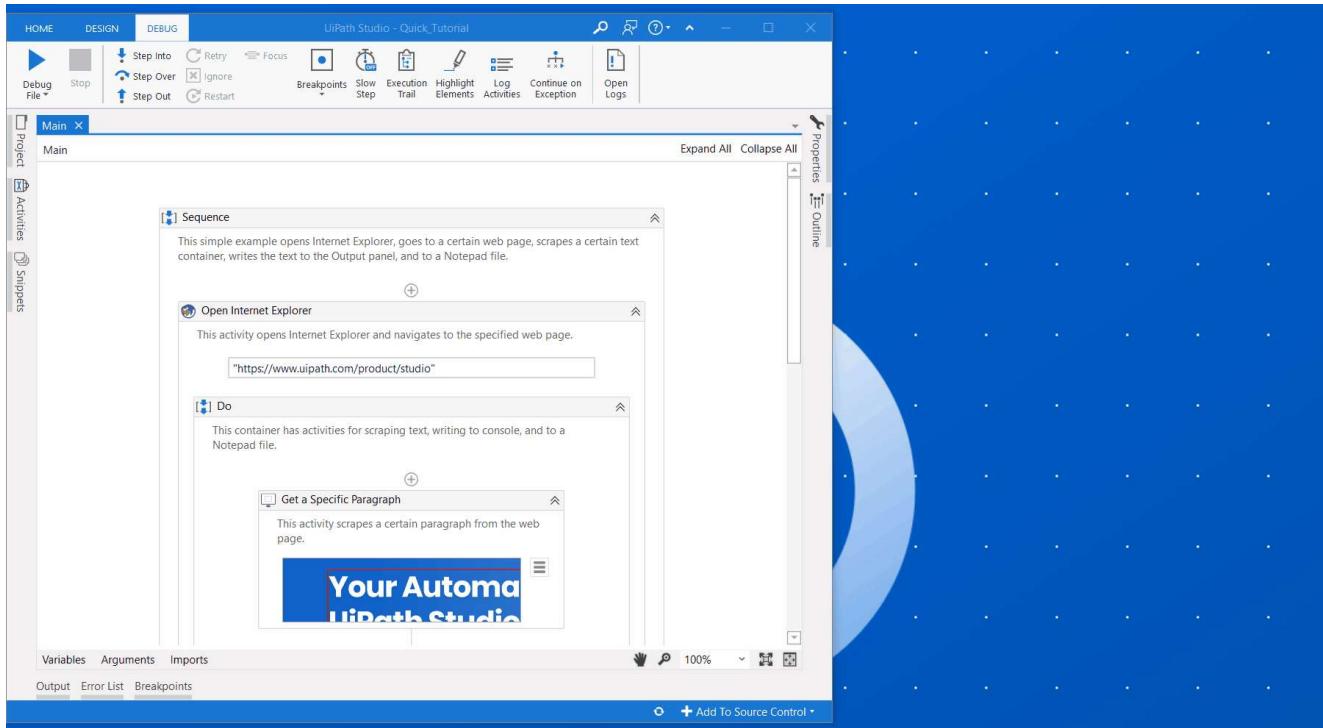
Although called **Slow Step**, the action comes with 4 different speeds. The selected speed step runs the debugging process slower than the previous one. For example, debugging with **Slow Step** at 1x runs it the slowest, and fastest at 4x. In other words, the speed dictates how fast the debugger jumps from one activity to the next.

Each time you click **Slow Step** the speed changes by one step. You can easily tell by the icon, which updates accordingly.

## Execution Trail

The **Execution Trail** ribbon button is disabled by default. When enabled, it shows the exact execution path at debugging. As the process is executed, each activity is highlighted and marked in the **Designer** panel, showing you the execution as it happens:

- executed activities are marked and highlighted in green;
- activities that were not executed are not marked in any way;
- activities that threw an exception are marked and highlighted in red.



## Highlight Elements

If enabled, UI elements are highlighted during debugging. The option can be used both with regular and step-by-step debugging.

## Log Activities

If enabled, debugged activities are displayed as Trace logs in the **Output** panel. Note that **Highlight Elements** and **Log Activities** options can only be toggled before debugging, and persist when reopening the automation project. This is not applicable for invoked workflows unless these files are opened in the **Designer** panel.

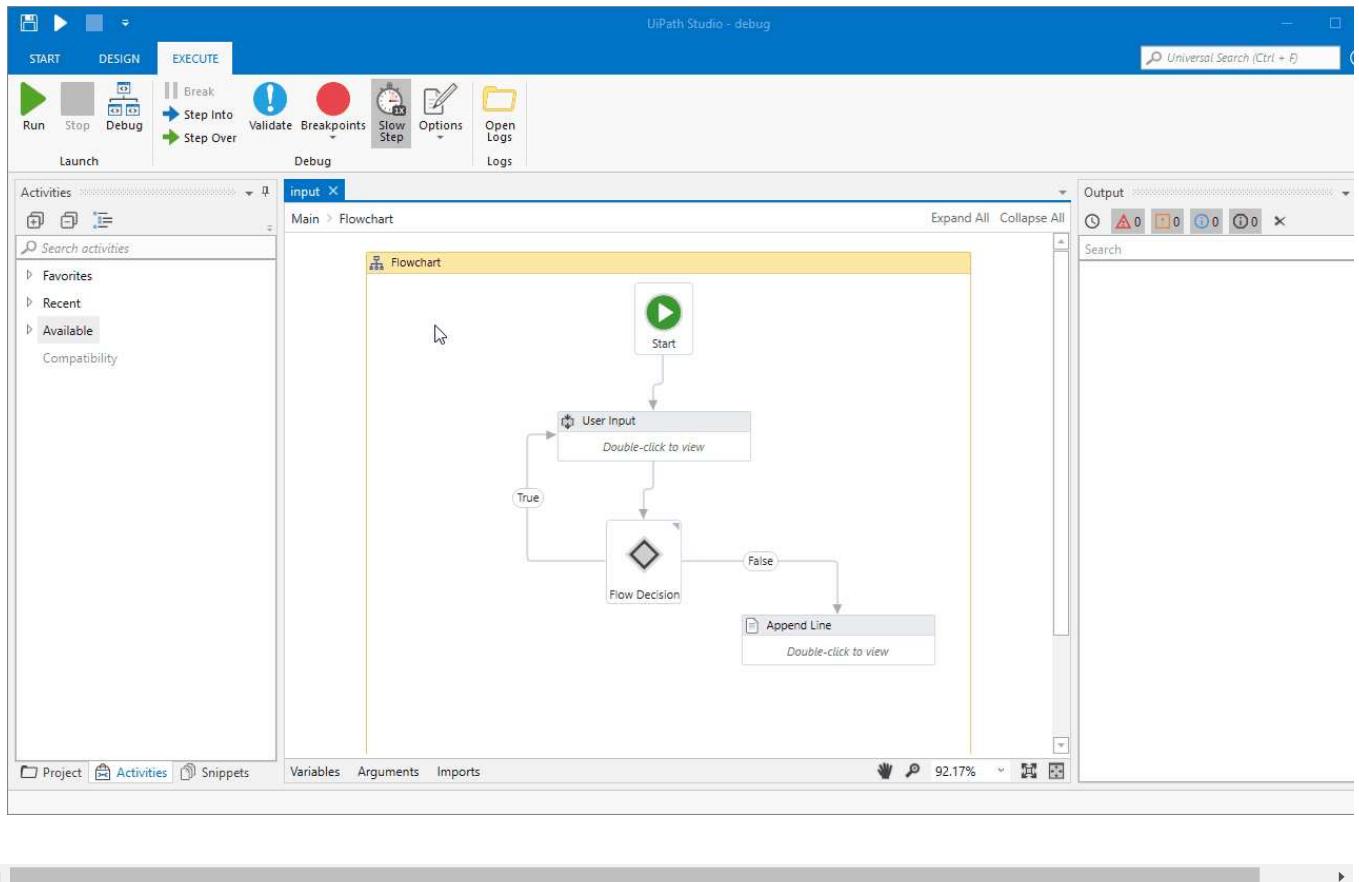
Logs are automatically sent to Orchestrator if connected, but you can have them stored locally by disabling the **Allow Development Logging** option from the **Robot Settings** tab in the Add or Edit user window.

Disabling **Log Activities** can be a way to send smaller log files to Orchestrator.

### **NOTE:**

When running processes from Studio, the logs sent to Orchestrator are always Trace when Log Activities is disabled, and Verbose when Log Activities is enabled. This overrides both the Robot and the Orchestrator setting.

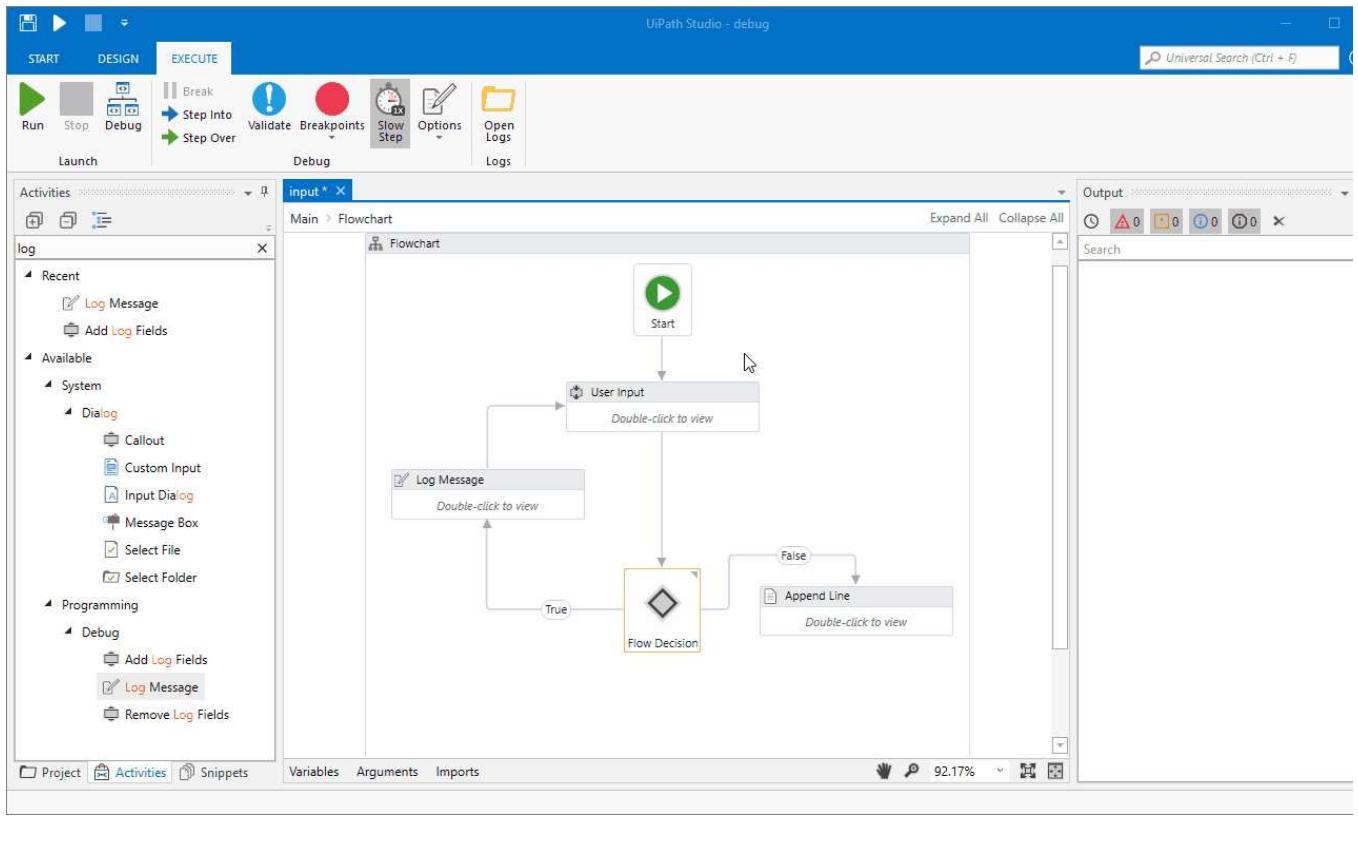
By default, the debugger logs activities so that each step appears in the **Output** panel. We recommend leaving it enabled for easier tracing, as you can see in the image below:



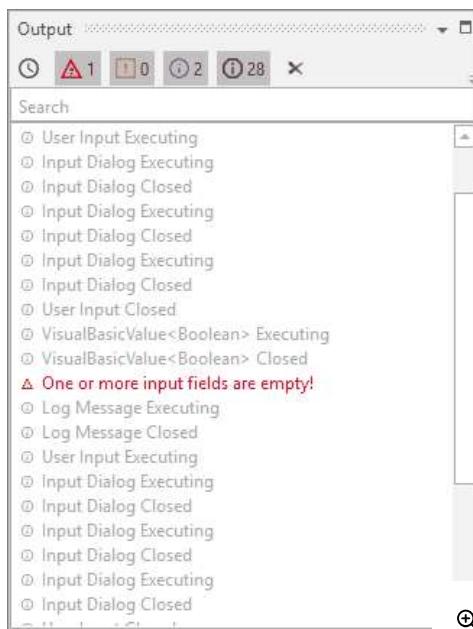
The issue here is that one or more input fields from the User Input sequence are blank, which is a True condition for the Flow Decision. You can tell this by the fact that, during debugging, the User Input sequence is executed twice, meaning that one or more fields were left blank during the first execution.

If you decide to disable the **Log Activities** option for debugging, Trace logs are not displayed in the **Output** panel. In the case of a normal execution with no errors, you only get to see the debug execution start and end times. However, adding a **Log Message** can help you determine where issues might occur.

For example, you can add a **Log Message** activity to inform you that, in this case, one or more input fields are empty. This message appears in the **Output** panel during debugging, even if the **Log Activities** option is disabled, as you can see below:



Remember that you can always filter the messages displayed in the [Output panel](#) by simply selecting the alert types of interest, or even clear all messages.



Note that by default all debugging logs are sent to Orchestrator. You can disable this by clearing the **Allow Development Logging** option from the [Settings tab](#) in the Add or Edit Robot window. If this option is disabled, debugging logs are only stored locally.

## Continue on Exception

This debugging feature is disabled by default. When disabled in the ribbon, it throws the execution error and stops the debugging, highlights the activity which threw the exception, and logs the exception in the [Output panel](#). If a [Global Exception Handler](#) was previously set in the

project, the exception is passed on to the handler.

When enabled, the exception is logged in the **Output** panel, the execution continues.

## Picture in Picture

The **Picture in Picture** ribbon option in the **Debug** tab is available for both executing and debugging processes or libraries in a separate session on your machine.

If enabled, whenever you select **Run** or **Run File**, **Debug** or **Debug File** the process starts either in a separate session or in a virtual desktop in the user session. If **Picture in Picture** is disabled, debugging and execution is performed in the current session.

Having the option to run a process in Picture in Picture (PiP) can be very useful in attended automation. Verify whether a process runs successfully in PiP, and then update the project settings to indicate if it can be executed using this feature after it is published:

1. In the **Project** panel, click **Settings**  to open the Project Settings window.

2. In the **General** tab:

- **PiP Options** - Indicate whether the project was tested using Picture in Picture and whether it should start in PiP by default.
  - **Tested for PiP usage; Starting in PiP** - The automation has been approved to run in PiP mode. When run, it starts in PiP by default.
  - **Tested for PiP usage; Not starting in PiP by default** - The automation has been approved to run in PiP mode. When run, it starts in the main session or desktop by default.
  - **Not tested for PiP usage** - The automation has not been approved to run in PiP mode. When run, it starts in the main session or desktop by default. If run in PiP, a dialog informs the user it was not tested using this feature and prompts for confirmation before proceeding.
- **PiP Type** - Select how to isolate the automation from the user session when running the project in PiP: **New Session** (child session on the machine) or **New Desktop** (virtual desktop in the user session).

For more information, including limitations of this feature, see [Picture in Picture](#) in the Robot Guide.

## Remote Debugging

When this feature is enabled, all run and debug operations are performed on a specified remote robot instead of the robot installed locally, allowing you to test the automation on different environments. For more information, see [Remote Debugging](#).

## Profile Execution

You can identify performance bottlenecks in the workflow when you debug the file. For more information, see [Profile Execution](#).

## Open Logs

Clicking **Open Logs** brings up the %localappdata%\UiPath\Logs folder where logs are locally stored. The naming format of log files is YYYY-DD-MM\_Component.log (such as 2018-09-12\_Execution.log, or 2018-09-12\_Studio.log). Read more about logging [here](#).



Default Theme

English

# Studio User Guide

RELEASE: 2023.4 

## TABLE OF CONTENTS

### [The Locals Panel](#)

## The Locals Panel

The **Locals** panel displays properties or activities and user-defined variables and arguments. The panel shows:

- **Exceptions** - the description and type of the exception.
- **Arguments**
- **Variables**
- **Properties of previously executed activity** - only input and output properties are displayed.
- **Properties of current activity**

The panel is only visible while debugging. Right-click an argument, variable or property of the currently executing activity to add it to the **Watch** panel and monitor its execution throughout the debugging process.

The **Arguments**, **Properties**, and **Variables** categories can be compressed or expanded. The same is available for complex objects, which are displayed in a tabular way.

Locals		
Name	Type	Value
<b>Arguments</b>		
FirstArgument	Int32	0
SecondArgument	String	"altceva"
ThirdArgument	String	null
<b>Properties of Read Range (previous)</b>		
Range	InArgument<String>	null
DataTable	OutArgument<DataTable>	[OrderDate,Region,Rep,Item,Units,Un
WorkbookPath	InArgument<String>	"read_range_example.xlsx"
Password	InArgument<String>	null
SheetName	InArgument<String>	"Sheet1"
<b>Variables</b>		
GetRange	DataTable	[OrderDate,Region,Rep,Item,Units,Un
FirstVariable	Int32	0
SecondVariable	Int32	0
Result	Int32	0

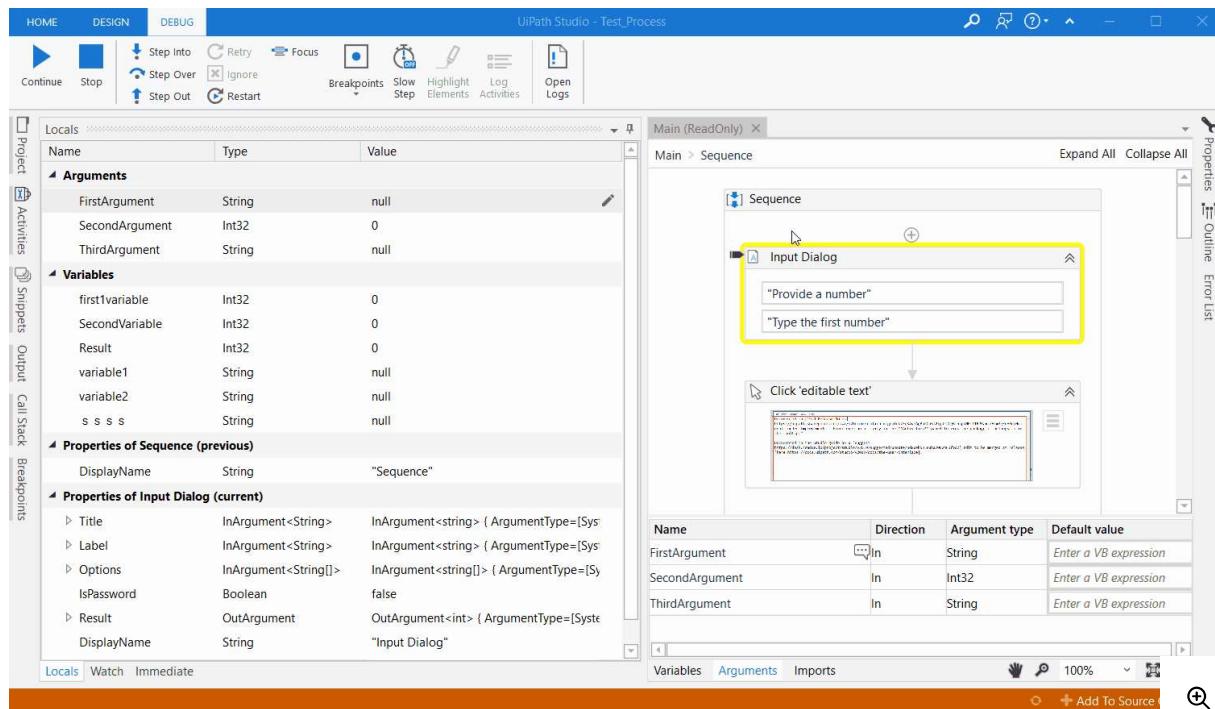
Property values are limited to a certain size. If a variable generates a large output, the value is truncated both in the **Locals** panel and in the **Immediate** panel.

When debugging is paused:

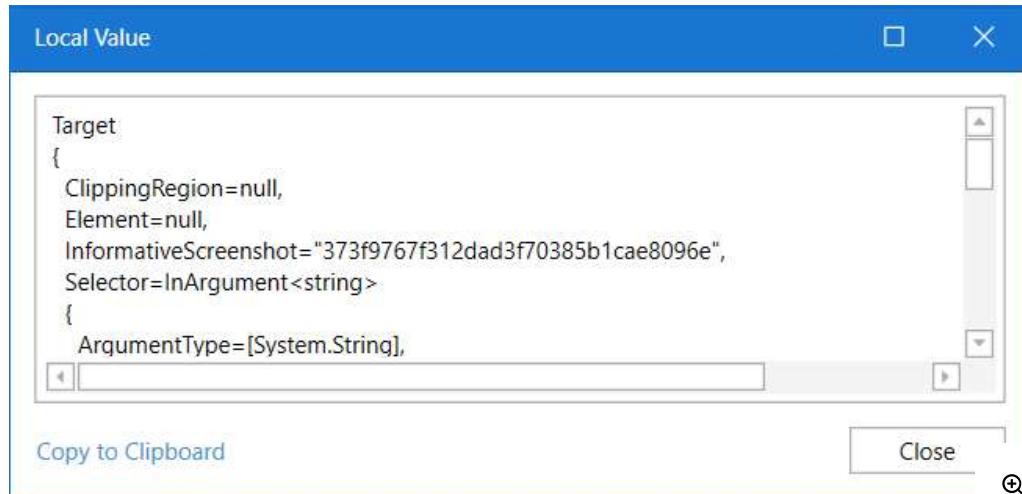
- You can modify the properties of the current activity and the values of variables and arguments by hovering over their **Value** field, clicking the  button next to them, and then making edits in the **Local Value** window.

 **NOTE:**

The edits made in the Local Value window are not saved in the file. After debugging is finished, the old values are still present in the Designer.



- You can inspect the values of other items in the **Locals** panel in detail by hovering over the **Value** field and clicking the button to open the **Local Value** window.



Upon clicking the **Copy to Clipboard** option, the information is copied to the clipboard.



Default Theme

English

# Studio User Guide

RELEASE:

2023.4



## TABLE OF CONTENTS

### [The Breakpoints Panel](#)

[Breakpoint Settings](#)

[Field Descriptions for Breakpoint Settings](#)

[The Breakpoints Panel](#)

[Context Menu for Breakpoints](#)

[Known Limitations in Windows-legacy Projects](#)

# The Breakpoints Panel

**Breakpoints** are used to purposely pause the debugging process on an activity which may trigger execution issues. Setting a condition and/or hit count turns the simple breakpoint to a conditional one. Adding logging results turns the conditional breakpoint in a conditional tracepoint. Adding only a logging message transforms the breakpoint to a simple tracepoint.

You can place and modify a breakpoint on any activity as follows:

- from the context menu, right-click an activity and select **Toggle Breakpoint**;
- by selecting the activity, and clicking the **Breakpoints** button on the **Debug** tab;
- by pressing F9 while the desired activity is selected.

A single activity needs to be selected for a breakpoint to be toggled. You can, however, toggle as many breakpoints as you see fit. Make sure that the order of activities in the workflow is not changed after the breakpoint is set.

Each breakpoint or tracepoint receives a specific icon based on its state. The icon is set on the activity and visible in the **Breakpoints** panel.

Breakpoints				
	Activity Name	File Path	Condition	Log Message
	Enabled Breakpoint	Main.xaml		
	Disabled Breakpoint	Main.xaml		
	Enabled Conditional Breakpoint	Main.xaml	FirstVariable=1	
	Disabled Conditional Breakpoint	Main.xaml	FirstVariable>1	
	Enabled Tracepoint	Sequence.xaml		ThirdVariable + "is higher than 0"
	Disabled Tracepoint	Sequence.xaml		FourthVariable + "is higher than 3"
	Enabled Conditional Tracepoint	Sequence.xaml	FifthVariable>6	FifthVariable + "is higher than 6"
	Disabled Conditional Tracepoint	Sequence.xaml	FourthVariable>3	FourthVariable + "is higher than 3"

Type	Description
<b>Breakpoints</b>	Breakpoints pause the debugging process before the activity is executed. Breakpoints can have the following states: <ul style="list-style-type: none"> <li>Enabled - </li> <li>Disabled - </li> </ul>
<b>Conditional Breakpoints</b>	Conditional breakpoints are breakpoints that depend on a set condition and/or a hit count. Conditional breakpoints can have the following states: <ul style="list-style-type: none"> <li>Enabled - </li> <li>Disabled - </li> </ul>
<b>Tracepoints</b>	Tracepoints are breakpoints with set logged messages. When the tracepoint is reached during debugging, the message is logged at trace level. Tracepoints can have the following states: <ul style="list-style-type: none"> <li>Enabled - </li> <li>Disabled - </li> </ul>
<b>Conditional Tracepoints</b>	Conditional tracepoints have a set condition or hit count, and a logged message. The message is logged when the condition is met the number of times stated in the hit count field. Conditional tracepoints can have the following states: <ul style="list-style-type: none"> <li>Enabled - </li> <li>Disabled - </li> </ul>

To modify the state of a breakpoint or tracepoint select the activity and press F9, click the icon in the **Breakpoints** panel, or use the **Designer** or **Breakpoints** panel context menus. You can also click the **Breakpoints** button on the **Debug** tab, open the drop-down menu and click **Toggle Breakpoint**.

#### NOTE:

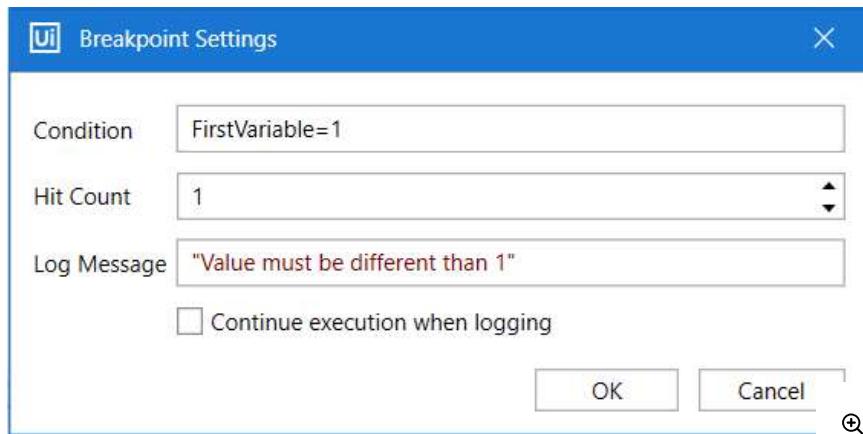
**Breakpoints** set during design time persist when reopening the automation project. Breakpoints don't persist at runtime, only at <https://docs.uipath.com/studio/standalone/2023.4/user-guide/the-breakpoints-panel>

debugging.

Select a breakpoint in the panel and click the **Delete** button to remove it. The **Delete all breakpoints** option enables you to delete all the breakpoints in the current project. The **Enable all breakpoints** option helps you enable all breakpoints in the currently opened project. Consequently, the **Disable all breakpoints** option disables all breakpoints. Multiple selection is available in the **Breakpoints** panel.

## Breakpoint Settings

The **Breakpoints** panel comes with a set of settings that can be individually adjusted for each toggled breakpoint part of the automation project. Click the  icon to open the window.



 **NOTE:**

Please take into consideration that any expression added in the **Condition** field is not validated.

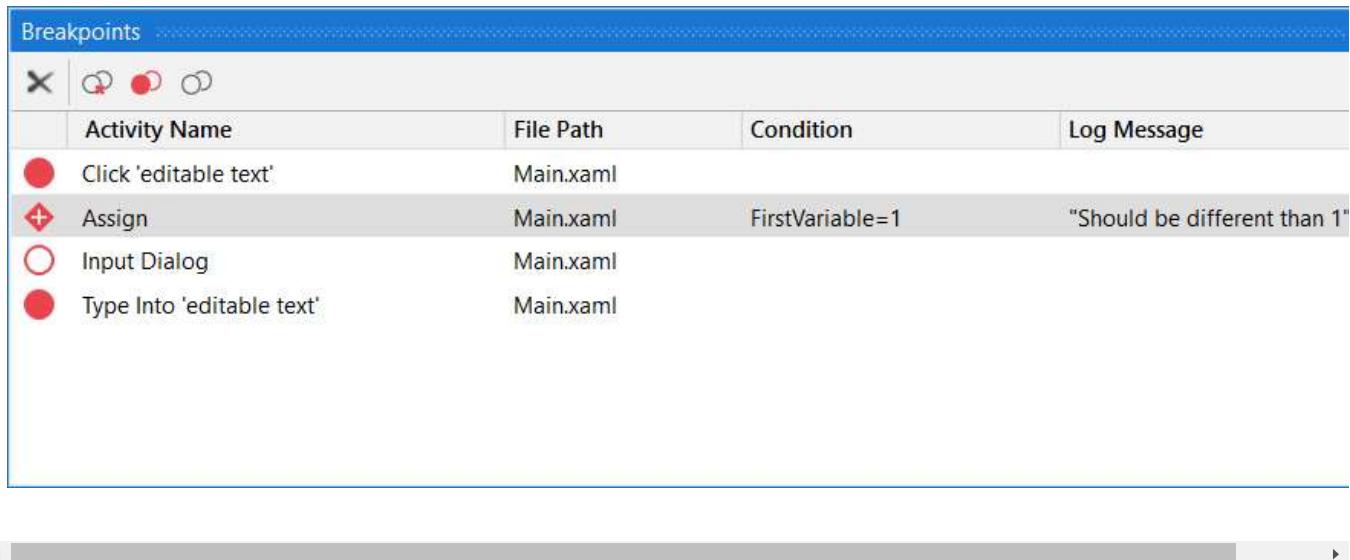
## Field Descriptions for Breakpoint Settings

The **Breakpoint Settings** window has the following options:

Option	Description
<b>Condition</b>	The condition for the breakpoint. If the condition is met during debugging, the execution breaks and the activity is highlighted.
<b>Hit Count</b>	Specifies the number of times the condition must be met before the execution breaks. If the hit count is higher than the number of times the condition can be met, the execution does not stop upon encountering the breakpoint. The maximum hit count value is 32,767.
<b>Log Message</b>	Specifies the message to be logged at trace level when the condition is met. The message is visible in the <b>Output</b> panel. If a condition is not set, the message is still logged.
<b>Continue execution when logging</b>	If selected, the execution is not paused when the condition is met and the specified message is logged. Available only if a log message was previously set.

Settings for any breakpoint in the project are visible upon hovering the breakpoint in the **Designer** panel.

# The Breakpoints Panel



The **Breakpoints** panel displays all breakpoints in the current project, together with the file in which they are contained. The **Activity Name** column shows the activity with the toggled breakpoint, while the **File Path** column displays the file and its location.

The **Condition** column displays conditions set to breakpoints. The **Log Message** column shows messages to be logged if the condition is met. Hover over the breakpoint tag on an activity to view its condition and log message.

Double-click on a breakpoint to see the activity highlighted in the **Designer** panel. Use context menu options or the **Breakpoints** button in the ribbon to enable or disable breakpoints.

To delete multiple breakpoints, select them and click **Delete** in the context menu, or the **Delete** button in the panel. This removes the breakpoints from the current file.

The **Delete all**, **Enable all** and **Disable all** breakpoints buttons perform actions on all breakpoints listed in the panel, regardless if they are selected or not.

## Context Menu for Breakpoints

Right-click an item in the **Breakpoints** panel to open the context menu with the following options:

Option	Description
<b>Delete</b>	Deletes the selected breakpoints.
<b>Focus</b>	Jumps to the breakpoint in the <b>Designer</b> panel.
<b>Enable</b>	Enables the selected breakpoints.
<b>Disable</b>	Disables the selected breakpoints.
<b>Settings</b>	Opens the <b>Breakpoint Settings</b> window for adding a condition to the breakpoint.

**NOTE:**

Breakpoint conditions are not evaluated when using **Validate File** or **Validate Project**.

# Known Limitations in Windows-legacy Projects

Execution does not pause when a breakpoint is hit if the workflow contains:

- A UI Automation activity where any option is selected for the **Wait for page load** property.
- Two variables of different types that are called using the same expression.



Default Theme

English



# Studio User Guide

RELEASE:

2023.4



## TABLE OF CONTENTS

[The Watch Panel](#)

## The Watch Panel

Similar to the **Call Stack** panel, the **Watch** panel is only visible during debugging. It can be set to display the values of variables or arguments, and values of user-defined expressions that are in scope. These values are updated after each activity execution while debugging.

Watch		
Expression	Value	Type
FirstVariable	8	Int32
Result	12	Int32
SecondVariable	4	Int32
FirstVariable/SecondVariable	2	Double

Add Watch

Call Stack   Watch   Breakpoints   

Column	Description
Expression	The variable, argument or expression to be monitored.
Value	The value of the variable, argument or expression.
Type	The type of variable, argument or expression.

Variables or arguments can be added to the **Watch** panel in the following ways:

- In the **Watch** panel, click the **Add Watch** field and type the name of the variable or argument;
- In the **Locals** panel, right-click a variable or argument and select **Add to Watch**;
- In the **Variables** or **Arguments** panel, right-click a variable or argument and select **Add Watch**.

To copy a value from the **Watch** panel, select the value, right-click and select **Copy**.

To remove one or more entries from the panel, right-click and select **Delete** or **Clear All**.



## Default Theme

English



# Studio User Guide

RELEASE:

2023.4



## TABLE OF CONTENTS

### [The Immediate Panel](#)

Context Menu for the Immediate Panel

## The Immediate Panel

The **Immediate** panel is only visible during debugging, and it can be used for inspecting data available at a certain point during debugging. It can evaluate variables, arguments, or statements. To do so, simply type the variable or argument name in the **Immediate** window and press Enter.

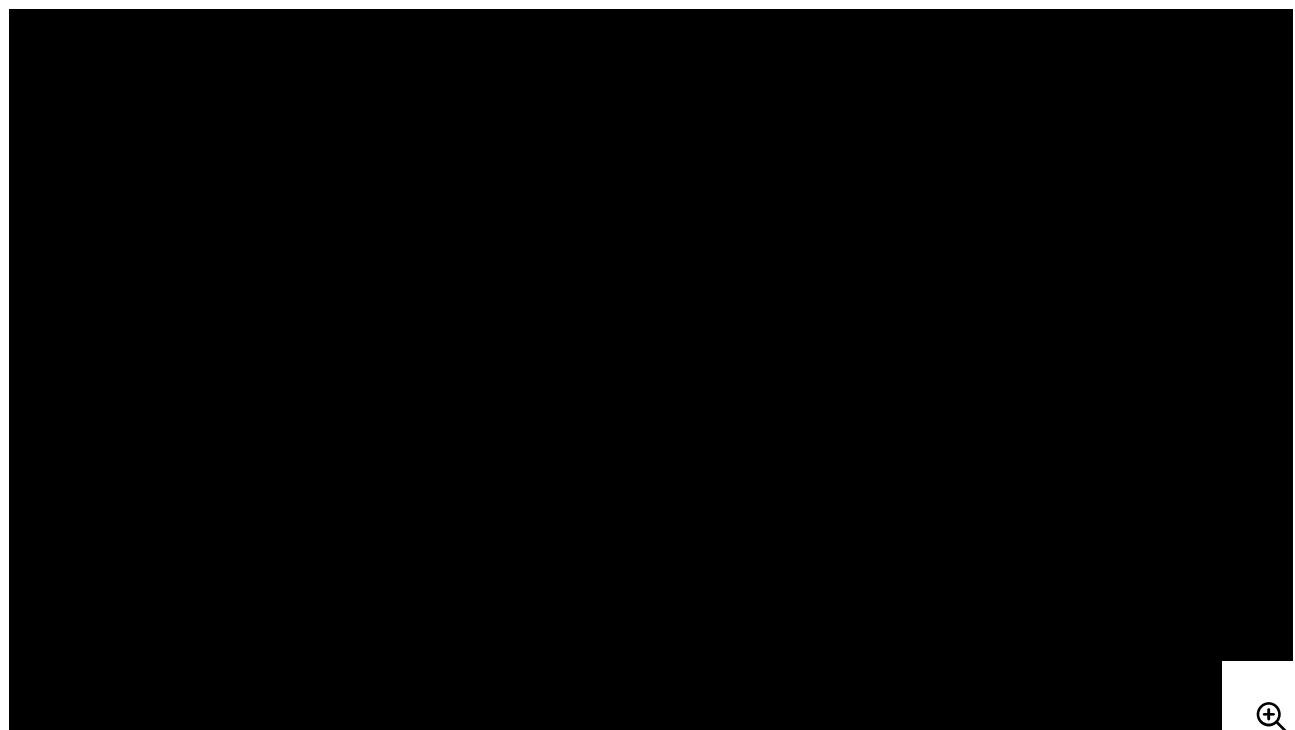
The Immediate panel displays a history of previously evaluated statements. The entries are:

- > FirstVariable  
↳ 32
- > FirstVariable=64  
↳ 64
- > FirstVariable.GetTypeCode  
↳ Int32
- > FirstVariable.CompareTo(SecondVariable).ToString  
↳ "1"

A horizontal scrollbar is visible at the bottom of the panel.

The **Immediate** panel keeps the history of previously evaluated statements, and they can be removed using the **Clear All** context menu option.

To remove a single line from the panel, select the text and press the **Space** key. When clicking inside a line and starting to type, the text is added to the input field.



Please take into consideration that the guidelines for [calling a function](#) apply to the **Immediate** panel as well, and parentheses should be used.

If you have a `List<string>` variable, it is recommended to use parentheses to view object-specific methods in the Intellisense window. For example, use `Names.First().ToUpper` instead of `Names.First.ToUpper` to capitalize the first element in a list of names.

## Context Menu for the Immediate Panel

Right-click in the **Immediate** panel to open the context menu with the following options:

Option	Description
<b>Copy</b>	Click <b>Copy</b> or use <b>Ctrl + C</b> to copy the selected text to clipboard.
<b>Clear All</b>	Use this option to clear all lines in the panel.
<b>Show IntelliPrompt</b>	Use <b>Ctrl + Space</b> to open IntelliPrompt.



Default Theme

English



# Studio User Guide

RELEASE:

2023.4



## TABLE OF CONTENTS

### [Remote Debugging](#)

[Remote Machine Connection](#)

[Prerequisites](#)

[Configuring the robot on the remote machine](#)

[Setting up a remote machine connection](#)

[Unattended Robot Connection](#)

[Prerequisites](#)

[Setting up an unattended robot connection](#)

[Remote Execution](#)

[Known limitations](#)

[Closing a Remote Debugging Connection](#)

# Remote Debugging

Automations may behave differently on different machines. If the machine on which an automation will run in production has a different configuration than the machine where it is designed (for example, the

machine has different hardware or software, different permissions, or it is in an isolated network) the process should be tested and debugged with the robot on that machine.

Remote debugging enables you to run and debug attended and unattended processes on robots deployed to remote machines, including on [Linux robots](#) that can run cross-platform projects.

You can connect to the remote robot using one of the following connection types:

- [Remote Machine](#) - Establish a TCP/IP connection to the robot on the remote machine.
- [Unattended Robot](#) - Connect to an unattended robot in the same tenant using Orchestrator.

Using remote debugging requires that the project is open in Studio. If you are using source control, to make sure you are working with the latest project version, we recommend enabling the [Enforce Check-In before Publish](#) design setting.

## Remote Machine Connection

To run or debug a project using a remote machine connection:

1. Make sure all the prerequisites are met.
2. On the remote machine, configure the robot to accept remote debugging requests.
3. If interactive authentication is enforced in Orchestrator and you want to run or debug an unattended process, make sure that the remote robot meets one of the following conditions:
  - Connected to Orchestrator using interactive sign-in.
  - Connected using client credentials or machine key and a user is also signed in from the Assistant.
  - Connected using client credentials or machine key, no user is signed in from the Assistant, and a troubleshooting session is enabled in Orchestrator for the machine. For more information, see [Debugging Unattended Processes](#).
4. In Studio:
  - a. [Set up a connection to the remote robot](#).
  - b. Make sure [remote execution is enabled](#).
  - c. Run or debug your project.

## Prerequisites

- TCP/IP connectivity exists between the Studio machine and the remote machine.
- The remote Robot is the same version as Studio.

## Configuring the robot on the remote machine

Before the remote robot can be used for debugging, the **UiPath.RemoteDebugging.Agent** utility on that machine must be configured to accept remote debugging requests from Studio:

### 1. Navigate to the installation directory:

- **For a Windows robot** - Open a command prompt in the UiPath installation folder (by default %PROGRAMFILES%\UiPath\Studio for per-machine installations, %localappdata%\Programs\UiPath\Studio for per-user installations).
- **For a Linux robot** - From a command line terminal, navigate to /root/application.
- **For a macOS robot** - From zsh, navigate to /Applications/UiPath Assistant.app/Contents/Robot.

### 2. Run the following command:

- **For a Windows robot** - UiPath.RemoteDebugging.Agent.exe enable --port <port\_number> --password <password> --verbose
- **For a Linux robot** - ~/application # dotnet ./UiPath.RemoteDebugging.Agent.dll enable --port <port\_number> --password <password> --verbose
- **For a macOS robot** - dotnet UiPath.RemoteDebugging.Agent.dll --port <port\_number> --password <password> --verbose

The arguments in the command are all optional:

- --port <port\_number> - Enter the port to use for receiving remote debugging commands from Studio. If no port is provided, the port **8573** is used by default.  
The port must be open in the firewall and not already bound by another application.
- --password <password> - Enter a password that must then be provided in Studio when setting up a connection to the remote debugging agent.
- --verbose - Log extra information to the console.

### 3. The following message is displayed:

```
Robot on machine <hostname> is waiting for remote debugging instructions on port  
<port_number>
```

### 4. Make a note of the **hostname** and **port\_number** values, they must be provided when setting up the connection in Studio.

No attended or unattended jobs can be executed from Orchestrator or from the local Assistant while the robot is in a remote debugging state. You can send remote debugging commands even to machines where the robot installation is not licensed.

## Setting up a remote machine connection

1. In Studio, select the **Debug** tab.
2. In the ribbon, select the arrow under **Remote Debugging**, and then select **Configure Remote Debugging** to open the **Remote Debugging Settings** window.
3. From the **Connection Type** dropdown, select **Remote Machine**.

4. Provide the following information in the corresponding boxes:

- **Host** - The hostname or IP address of the remote machine.
- **Port** - The port to use. The default port is **8573**.
- **Password** - The password provided when the remote debugging agent was configured on the robot machine, if applicable.

5. (*Optional*) To make sure a connection can be established with the current setup, click **Test Connection**.

6. Click **Save**.

The screenshot shows the 'Remote Debugging Settings' dialog box. At the top, there's a blue header bar with the title 'Remote Debugging Settings'. Below it, the main title 'Remote Debugging Settings' is displayed, followed by a subtitle: 'Configure the target for remote execution on a Remote Machine or an Unattended Robot connected to Orchestrator.' A 'More Info' link is also present. The form contains several input fields:

- 'Connection Type \*': A dropdown menu showing 'Remote Machine'.
- 'Host \*': An input field containing a blurred IP address.
- 'Port \*': An input field containing '8573'.
- 'Password': An input field with a blurred password.

At the bottom of the dialog are two buttons: 'Save' and 'Cancel'. There's also a navigation bar at the very bottom with left and right arrows.

## Unattended Robot Connection

To run or debug a project remotely using an unattended robot connected to Orchestrator:

1. Make sure all the [prerequisites](#) are met.
2. Set up a connection to the remote robot.
3. Make sure [remote execution is enabled](#).
4. Run or debug your project.

## Prerequisites

- Studio and the target robot are connected to the same Orchestrator tenant.
- Studio, the target robot, and Orchestrator are running version 2021.10 or later. For the robot, 2021.10.6 is the minimum version required for running projects from Studio versions starting with 2021.10.6.
- The user signed in to Studio has permissions to start jobs, and to create and delete storage buckets and storage files in the same folder context as the target robot. In addition, the robot account must have view permissions for storage buckets and storage files.
- The unattended robot is configured and the machine has one of the following runtime licenses available: **Unattended**, **NonProduction**, **Testing**.

 **NOTE:**

Testing runtimes for remote debugging are supported in Orchestrator 2022.4 and later.

- For debugging foreground processes, the option **Run foreground automations** is enabled for the robot in Orchestrator.
- If a troubleshooting session is used, it must be enabled only after connecting the robot to the Orchestrator.

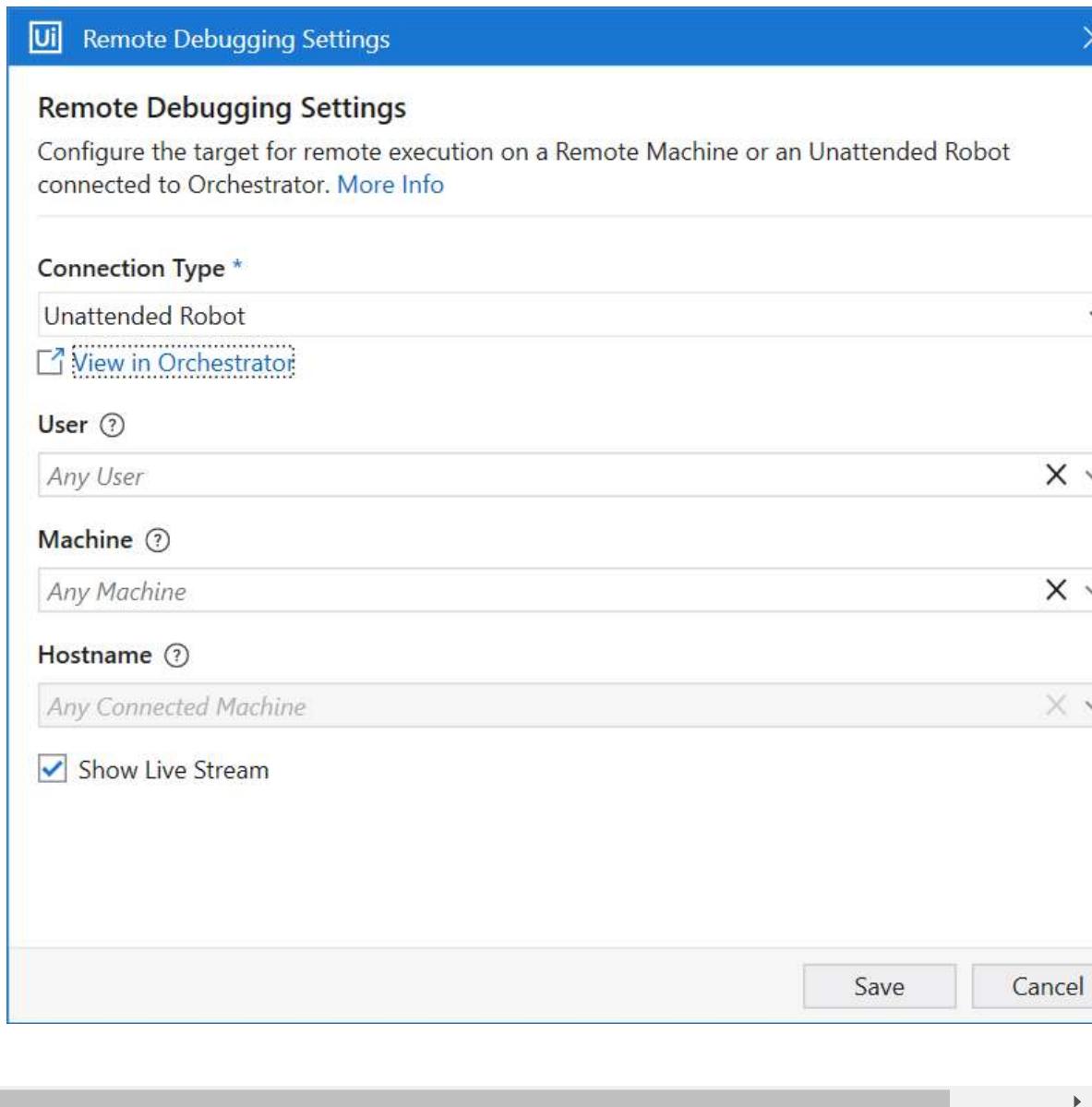
## Setting up an unattended robot connection

1. In Studio, select the **Debug** tab.
2. In the ribbon, select the arrow under **Remote Debugging**, and then select **Remote Debugging Settings**.
3. From the **Connection Type** dropdown, select **Unattended Robot**.
4. To use any connected machine that is available in the Orchestrator folder [selected from the Studio status bar](#), click **Save**. If you want to select the machine to connect to, use the following options:
  - **User** - Select an account with an unattended robot assigned to the Orchestrator folder.
  - **Machine** - Select a machine or template from the Orchestrator folder.
  - **Hostname** - Select a machine from the list of connected machines.

- **Show Live Stream** - Select to see the actions performed by the robot. For more information, see [Live Streaming and remote control](#).

**NOTE:**

If changes are made to the account setup in Orchestrator, refresh the Orchestrator connection using the button in the Studio status bar so that they are reflected in this window.



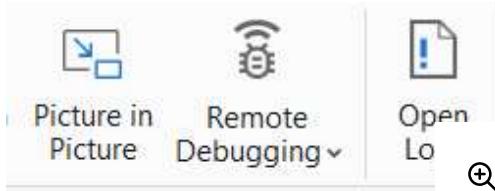
## Remote Execution

When a remote debugging connection is established, clicking the **Remote Debugging** button in the ribbon toggles between remote and local execution. Before you select a run or debug operation, make sure the desired option is enabled (remote or local).

- As long as the button is highlighted in gray, all run and debug operations (**Debug File**, **Run File**, **Debug Project**, **Run Project**, **Step Into/Over/Out**, **Test Activity**, **Run to/from this activity**) are performed on the remote robot.



- As long as the button is not highlighted in gray, all run and debug operations are performed on the local robot.



The remote debugging experience is similar to the local debugging experience and all the features available for local debugging are also available for remote debugging. When remote execution that was triggered from the Debug tab is in progress, the Studio status bar is colored in **green**.

Depending on the type of connection used for remote debugging, the remote robot gets the activity packages required to execute a project as follows:

- Remote machine connection** - Studio sends the list of project dependencies and activity feeds (package sources) to the remote robot, which uses the feeds provided by Studio to download the required packages.
- Unattended robot connection** - Studio sends only list of project dependencies to the remote robot, which uses the Orchestrator feeds and the activity feeds configured on the remote robot to download the required packages.

For an unattended robot connection, selecting **Show Live Stream** lets you visualize the actions that the robot is performing in real time. During remote execution, a new live stream window opens, which you can move and resize. You can also take remote control without pausing the automation for in-depth debugging or if the execution is blocked (for example, by an UI element that needs to be clicked). The live stream window disappears as soon as the workflow ends. While no additional actions are needed for Serverless Automation Cloud Robots, Windows robots require [setting up a VNC server](#).

## Known limitations

- When you use a remote machine connection, if you pause debugging for an extended period of time, a *Connection Closed* error might occur in Studio even though on the remote machine the connection still appears to be active. To avoid this issue, you can increase the TCP idle timeout in your cloud or on-premises load balancer.
- Remote debugging long-running workflows is only supported for unattended robot connections.
- When you use an unattended robot connection, selecting the **Picture in Picture** option does not start execution in a separate session.
- The **Show Live Stream** option works only if the robot service on the unattended robot is deployed in service mode. For more information, see [Robot Service](#).
- The **Show Live Stream** option is only supported for Automation Cloud Orchestrator and Automation Suite Orchestrator.

## Closing a Remote Debugging Connection

To disable the remote debugging connection, when no debugging execution is in progress, open the **Remote Debugging Settings** window, set the **Connection Type** to **Disabled**, and then select **Save**.



Default Theme

English

# Studio User Guide

RELEASE: 2023.4

## TABLE OF CONTENTS

### [Profile Execution](#)

- [Overview](#)
- [How it works](#)
- [Considerations](#)
- [Analyzing profiling results](#)
- [Import profiling session](#)
- [Context menu for profiling](#)
- [Related articles](#)

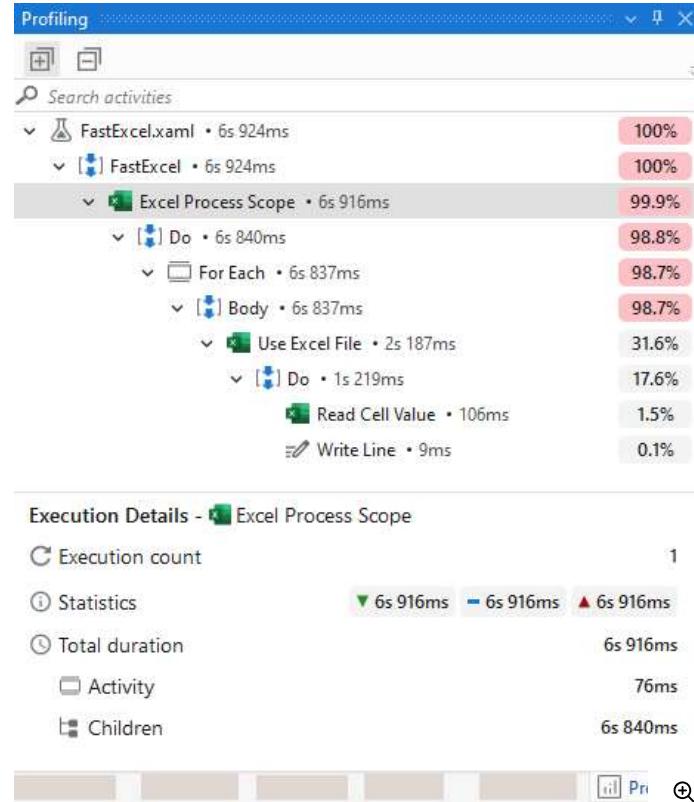
# Profile Execution

## Overview

Use Profile Execution to identify performance issues in workflow executions.

## How it works

When you run or debug a workflow, Profile Execution provides a performance analysis of all the operations, showing you a cumulative percentage of the execution time of each activity.



## Considerations

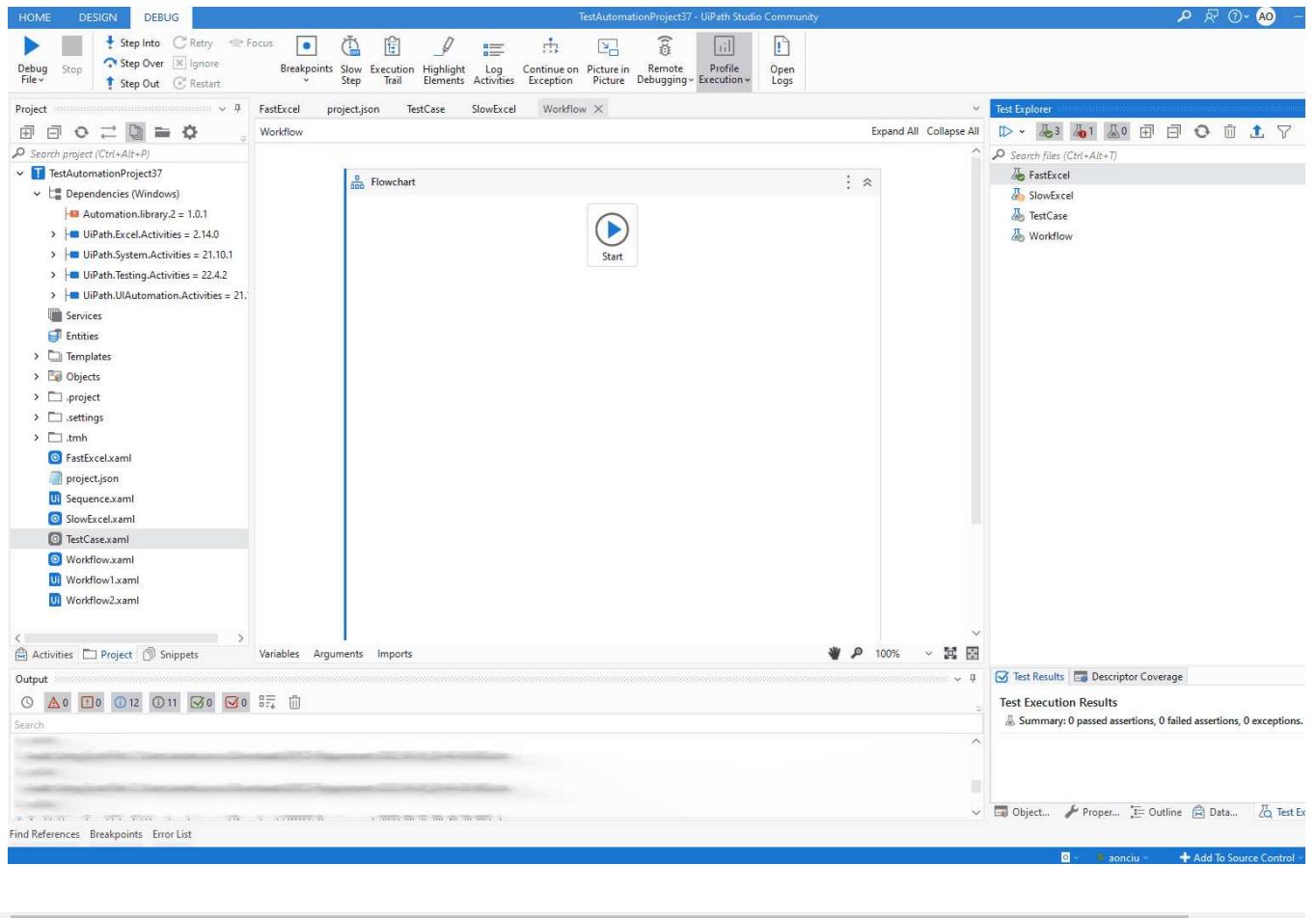
- Profiling data generated during [debugging](#) might be different from data generated during production (running the file).
- The data is stored for each session in C:\Users\username\AppData\Local\UiPath\ProfiledRuns (see [Import previous profiling sessions](#)).

## Analyzing profiling results

To profile an execution, run a file or [debug the file](#), then go to the **Debug** ribbon tab.

Make sure that your automation works correctly without slow performing workflows. If you identify potential flow issues, you can review workflows that take longer to be completed. Use the [context menu for profiling](#) to take actions.

The **Execution Details** at the bottom of the **Profiling** tab shows the number of executions and descriptive statistics such as the average duration and the minimum and maximum duration values.



## Import profiling session

You can import profiling sessions to examine previous runs. Focus is disabled on imported profiling sessions.

## Context menu for profiling

Action	Description
Open	Right-click the parent file in the Profiling tab and select <b>Open</b> to jump to the selected workflow.
Focus	Right-click an activity and select <b>Focus</b> to center the <b>Designer</b> panel view on the selected activity.
Search	Use the search function to look for specific activities.
Expand/Collapse All	Use <b>Expand All</b> and <b>Collapse All</b> to bring to view or collapse all activities.

## Related articles

[Debugging Actions](#)



Default Theme

English

# Orchestrator Standalone User Guide

DELIVERY: AUTOMATION CLOUD AUTOMATION SUITE STANDALONERELEASE: 2023.4 

## TABLE OF CONTENTS

### [Queue Item Statuses](#)

[Item Statuses](#)[Revision Statuses](#)[Statuses Diagram](#)

# Queue Item Statuses

Queue items can have two types of statuses:

1. Item Statuses
2. Revision Statuses

## Item Statuses

These statuses let you know if an item has been processed or not, and the stage of the process at a particular time. Item statuses are displayed in the **Status** column, in the **Transactions** page. Queue items can go through the following statuses:

- **New** – the item has just been added to the queue with the **Add Queue Item** activity, or the item was postponed, or a deadline was added to it, or the item was added after an attempt and failure of a previous queue item with auto-retry enabled.
- **In Progress** – the item was processed with the **Get Transaction Item** or the **Add Transaction Item** activity; when an item has this status, your custom progress status is also displayed, in the **Progress** column;
- **Failed** – the item did not meet a business or application requirement within the project and was therefore sent to a **Set Transaction Status** activity, which changed its status to Failed;
- **Successful** – the item was processed and sent to a **Set Transaction Status** activity, which changed its status to Successful;
- **Abandoned** – the item remained in the **In Progress** status for a long period of time (approx. 24 hours) without being processed;
- **Retried** – the item failed with an application exception and was retried. After the Robot finishes retrying the item, the status changes to Failed or Successful, according to your workflow.
- **Deleted** – the item has been manually selected from the **Transactions** page and marked as deleted; an item with this status can no longer be processed.

**⚠️ IMPORTANT:**

To support our effort of consolidating queue item final statuses, you can no longer use the `SetTransactionResult` endpoint to:

- change a transaction payload once it reaches a final state (be it **Failed** or **Successful**)
- rerun a transaction by using the `DeferDate` and `DueDate` properties, in order to move it out of a final state (be it **Failed**, **Successful**, **Abandoned**, or **Deleted**)

## Revision Statuses

These statuses let you perform version control but **only of queue items that have been abandoned or have failed with an application or business exception**. These statuses have to be manually set per item, by an assigned reviewer. All changes are tracked in the **History** tab of the **Audit Details** window. The reviewer can be assigned only when the item status is failed or abandoned, and reviewers cannot be changed after a revision status was added to the item. Only logged in reviewers can see requests assigned to them in the **Review Requests** page. Moreover, queue items can be assigned for revision in bulk.

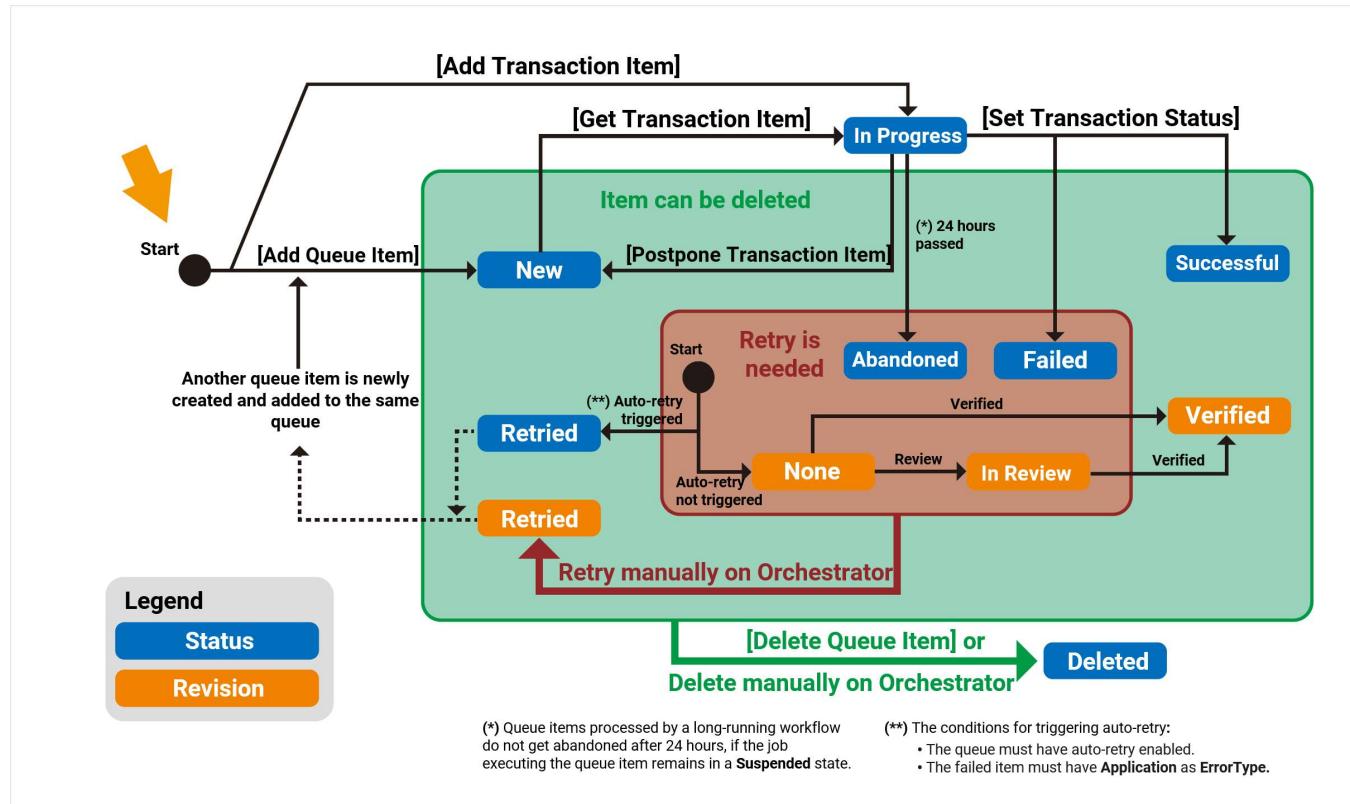
DETAILS		COMMENTS		HISTORY		
ACTION	WHO	STATUS	REVIEW	REVIEWER	TIME	C
Edit	admin	Failed	None	Documentation	29 minutes ago	
Edit	Documentation	Failed	None	vlad.tudoran	2 minutes ago	
Edit	Documentation	Failed	None		a few seconds ago	
Items: 10		◀ < Page 1/1 > ▶				3 items
						CLOSE

The following statuses are available:

- None** - this is the default status. It is set to all items, even if they failed or not.
- In Review** - a user has marked an item that has failed with app exception as in the process of being reviewed. This status does not have other implications in Orchestrator or Studio than changing the value in the **Revision** column on the **Queues** page.
- Verified** - a user has marked an item as verified. Items cannot be retried after the user sets this status. There are no other implications in Orchestrator or Studio than changing the value in the **Revision** column on the **Queues** page.
- Retried** - the item has been marked manually for retry. As a result, a new queue item with the **New** status is created. This is displayed in the **Items Details** window of the indicated transaction.

## Statuses Diagram

The diagram below explains the queue items transition through each status.



## Default Theme

English

# Productivity Activities

## TABLE OF CONTENTS

### [About the Excel Activities Package](#)

Modern Activities

Workbook Activities

App Integration Activities

Troubleshooting and Limitations

## About the Excel Activities Package

The Excel activities package aids users to automate all aspects of Microsoft Excel, as we know it is an application intensely used by many in all types of businesses.

It contains activities that enable you to read information from a cell, columns, rows or ranges, write to other spreadsheets or workbooks, execute macros, and even extract formulas. You can also sort data, color code it or append additional information.

 **NOTE:**

The **UiPath.Excel.Activities** pack is compatible with the following **Microsoft Excel** versions:

- 2013
- 2016
- 2019
- Office 365

## Modern Activities

The activities grouped under **Modern** are the newest activities that were initially designed for the StudioX profile.

The **Excel Modern** design experience and activities provide the same functionality as the StudioX business activities by default in Studio. To find out more about working with Excel activities in StudioX, see [Excel Automation](#) in the StudioX guide.

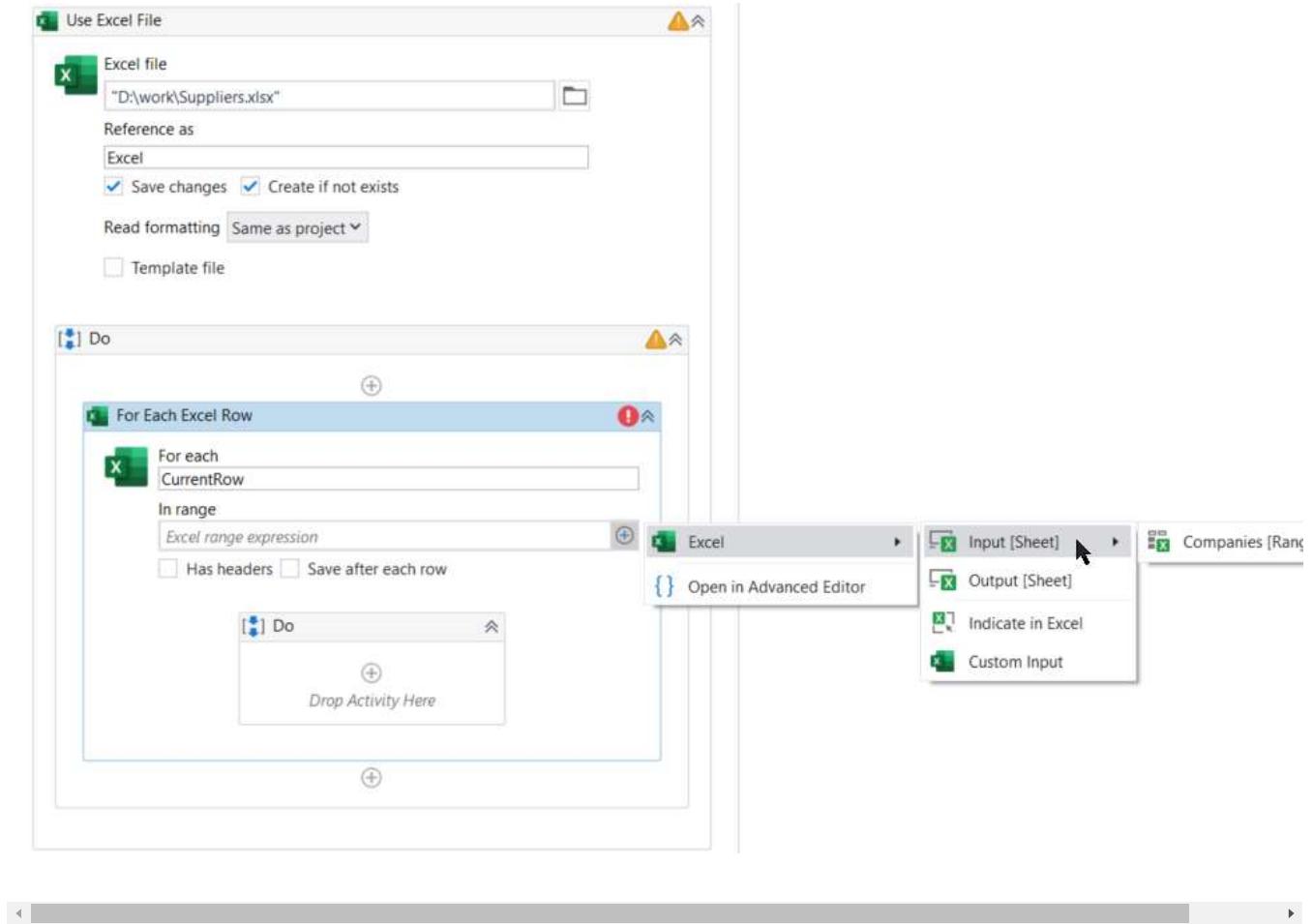
### Modern Design Experience in Studio

Starting with Studio v2021.10 and UiPath.Excel.Activities v2.11.0, a modern design experience is available in the Studio profile. In the modern experience, the modern activities replace the classic activities in the Studio profile, and the StudioX design experience is brought to Studio.

The modern design experience is enabled by default. You can select the design experience for each project from the Excel activities [project settings](#). When the classic experience is enabled for a project, the [classic app integration activities](#) that do not support interacting with Excel from the Plus  menu are available in the Activities Panel in Studio instead of the modern activities.

Add the [Use Excel File](#) scope activity to indicate the file to automate and configure its child activities by selecting Excel data from the Plus  menu of each property without having to manually enter expressions:

- Browse the contents of the file from the menu and select data that matches the type of each property. For example, you can select cells, ranges, tables, sheets, or charts from the worksheet. You can also indicate the current row or current sheet in an iteration.



- If you find it easier to work in Excel, use the **Indicate in Excel** option to select data directly from the file. This functionality requires that you install the [Excel Add-in](#).

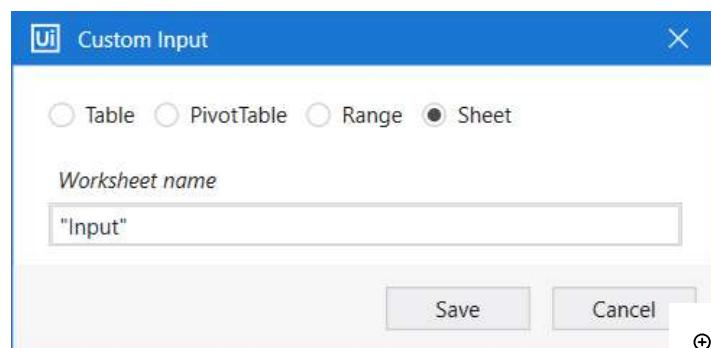
A1

A	B	C	D	E	F	G	H	
1	Id	Internal Name	External Name	Website	Industry	Recurrent Supplier	Supplier Since	Number of E
2	1	VPN Services	Secure VPN Provider	wwwvpn.com	Communications	TRUE	36578	
3	2	Energy Provider	ENEL	www.enel.it	Energy	TRUE	32926	
4	3	HR Provider	HR Expert	www.hrexpert.com	Human Resources	TRUE	40219	
5	4	HR Provider	HR Top Recruiters	www.hrtoprecruiters.com	Human Resources	TRUE	42776	
6	5	Internet Provider	InterPower Group	www.interpowergroup.com	Telecom	TRUE	40219	
7	6	Bank	Smart Bank	www.smartbank.com	Banking	TRUE	40413	
8	7	Wood Provider	Creative Wood	www.creativewood.com	Wood	TRUE	43506	
9	8	Metal Provider	Modern Metal	www.modernmetal.com	Metal	TRUE	43506	
10	9	Food Provider	Hungry	www.hungryfood.com	Food	TRUE	43141	
11	10	Vegetables Provider	Angry Vegetables	www.angryvegetables.com	Agriculture	TRUE	43138	
12	11	Trainers	Expert Trainers	www.experttrainers.com	Education	TRUE	43138	
13	12	Contractors 1	Teams	www.temps.com	Services	TRUE	43138	
14	13	Trainers 2	Technical Trainers	www.tehnicaltrainers.com	Education	TRUE	43138	
15	14	Trainers 3	Teoretic Trainers	www.teoretictrainers.com	Education	TRUE	43138	
16	15	Courses	Pragmatic	www.pragmatic.com	Education	TRUE	43138	
17	16	Contractors 3	Zodiac	www.zodiac.com	Services	TRUE	42558	

Input   Output   +

Ready   Average: 20184.94444   Count: 70   Sum: 363329

- Use the **Custom Input** option to manually specify the input based on table, chart, or sheet names or to enter references to a cell or a range.



If the file you want to automate does not exist yet, you can still use the options in the Plus menu by defining a file with the same structure as a template in the Use Excel File activity.

## Workbook Activities

[Workbook activities](#) can be executed even if Microsoft Excel is not installed on the machine and can only read/write data to and from the file.

# App Integration Activities

Classic app integration activities require the Excel app to be installed on the machine on which they run. For large and complex spreadsheets that span over a great number of rows and columns, we recommend using the App Integration activities, as these activities offer the best performance and consistency.

All app integration activities with the exception of CSV activities must be included in [Excel Application Scope](#) to work. Starting with version **2.10.4** of the Excel activities pack, these activities can also be included in the [Use Excel File](#) activity.

Dedicated activities enable you to work with .csv files, while all the others work with .xlsx and .xls. The ones that only work with [Excel Application Scope](#) can also work with .xlsm files.

## Troubleshooting and Limitations

### NOTE:

The old [Excel Binary File Format \(.xls\)](#) specific to Microsoft Excel 95 is not supported by the Excel Activities Pack.

- When you use activities that read, copy, or append ranges, please note that if a cell contains formatting information, Excel considers that cell as containing data, even though it appears empty.
- Please note that the Microsoft Office 2016 version 1708 (Build 8431.2079) changes the window title of files that are being edited and saved from "FileName.xlsx - Excel" to "FileName.xlsx - Saved". After closing the file, the filename reverts to normal.

This may cause some problems if your automation project uses selectors which contain the window title. For example, the following selector no longer works if your Excel version is the previously mentioned one: <wnd app='excel.exe' cls='XLMAIN' title='FileName.xlsx - Excel' /> <wnd cls='EXCEL7' title='FileName.xlsx' />. To prevent this issue, we recommend using a dynamic variable selector.

- Some Excel versions always open SharePoint documents in read-only mode (e.g. Excel 2019). You can check if a remote document supports editing by opening it manually in Excel on your local machine.



Default Theme

English



# Productivity Activities

## TABLE OF CONTENTS

### [Modern Activities](#)

## Modern Activities

This section includes activities found in the **UiPath.Excel.Activities** package that are designed for use in StudioX projects.

To find out more about working with Excel activities in StudioX, see [Excel Automation](#) in the StudioX guide.

[Use Excel File](#)

[Append Range](#)

[Auto Fill](#)

[Autofit Range](#)

[Change Pivot Data Source](#)

[Clear Sheet/Range/Table](#)

[Copy/Paste Range](#)

[Create Pivot Table](#)[Delete Column](#)[Delete Rows](#)[Delete Sheet](#)[Duplicate Sheet](#)[Export to CSV](#)[Fill Range](#)[Filter](#)[Find First/Last Data Row](#)[Find/Replace Value](#)[For Each Excel Row](#)[For Each Excel Sheet](#)[Format As Table](#)[Format Cells](#)[Get Excel Chart](#)[Insert Column](#)[Insert Chart](#)[Insert Rows](#)[Insert Sheet](#)[Lookup](#)[Match Function](#)[Protect Sheet](#)[Read Cell Formula](#)[Read Cell Value](#)[Read Range](#)[Refresh Excel Data Connections](#)[Refresh Pivot Table](#)

[Remove Duplicates](#)[Rename Sheet](#)[Run Spreadsheet Macro](#)[Save Excel File](#)[Save Excel File As](#)[Save Excel File As PDF](#)[Sort Range](#)[Text to Columns](#)[Unprotect Sheet](#)[Update Excel Chart](#)[VLookup](#)[Write Cell](#)[Write DataTable to Excel](#)

Default Theme

English



# Productivity Activities

## TABLE OF CONTENTS

---

### [Classic Activities](#)

## Classic Activities

This section includes classic (app integration) activities found in the **UiPath.Excel.Activities** package.

[Excel Application Scope](#)

[Append To CSV](#)

[Read CSV](#)

[Write CSV](#)

[Delete Column](#)

[Filter Table](#)

[Get Table Range](#)

[Insert Column](#)

[Sort Table](#)[Append Range](#)[Close Workbook](#)[Get Cell Color](#)[Read Cell](#)[Read Cell Formula](#)[Read Column](#)[Read Range](#)[Read Row](#)[Select Range](#)[Set Range Color](#)[Write Cell](#)[Write Range](#)[Save Workbook](#)[Create Table](#)[Get Workbook Sheet](#)[Get Workbook Sheets](#)[Refresh Pivot Table](#)[Create Pivot Table](#)[Get Selected Range](#)[Copy Sheet](#)[Delete Range](#)[Auto Fill Range](#)[Copy Paste Range](#)[Execute Macro](#)[Insert/Delete Columns](#)[Insert/Delete Rows](#)

[Invoke VBA](#)[LookUp Range](#)[Remove Duplicates Range](#)[Excel Process Scope](#)

Default Theme

English



# UI Automation Activities

## TABLE OF CONTENTS

### [Anchor Base](#)

#### Properties

Example of using the Anchor Base activity

## Anchor Base

`UiPath.Core.Activities.AnchorBase`

A container that searches for a UI element by using other UI elements as anchors. This should be used when a reliable selector is not available.

 **NOTE:**

For reliability reasons, it is recommended to avoid the usage of the `idx` attribute in selectors. In such cases, consider adding information about ancestor elements, using [relative elements](#) or the **Anchor Base** activity.

## Properties

### Input

- **AnchorPosition** - Specifies to which edge of the container the UI element is anchored. The following options are available:
  - **Auto** - If selected, the Bottom case is not considered. Searches for the UI element to the left, right or bottom of the container. The closest one is selected. If multiple elements are found at equal distance from the anchor, the target is selected in this order: Right, Left, Bottom.
  - **Left** - Searches for the UI element to the right of the container.
  - **Top** - Searches for the UI element at the bottom of the container.
  - **Right** - Searches for the UI element to the left of the container.
  - **Bottom** - Searches for the UI element at the top of the container.
  - **OnTop** - Searches for an UI element that has a label on it.

## Common

- **DisplayName** - The display name of the activity.
- **ContinueOnError** - Specifies if the automation should continue even when the activity throws an error. This field only supports Boolean values (True, False). The default value is False. As a result, if the field is blank and an error is thrown, the execution of the project stops. If the value is set to True, the execution of the project continues regardless of any error.

 **NOTE:**

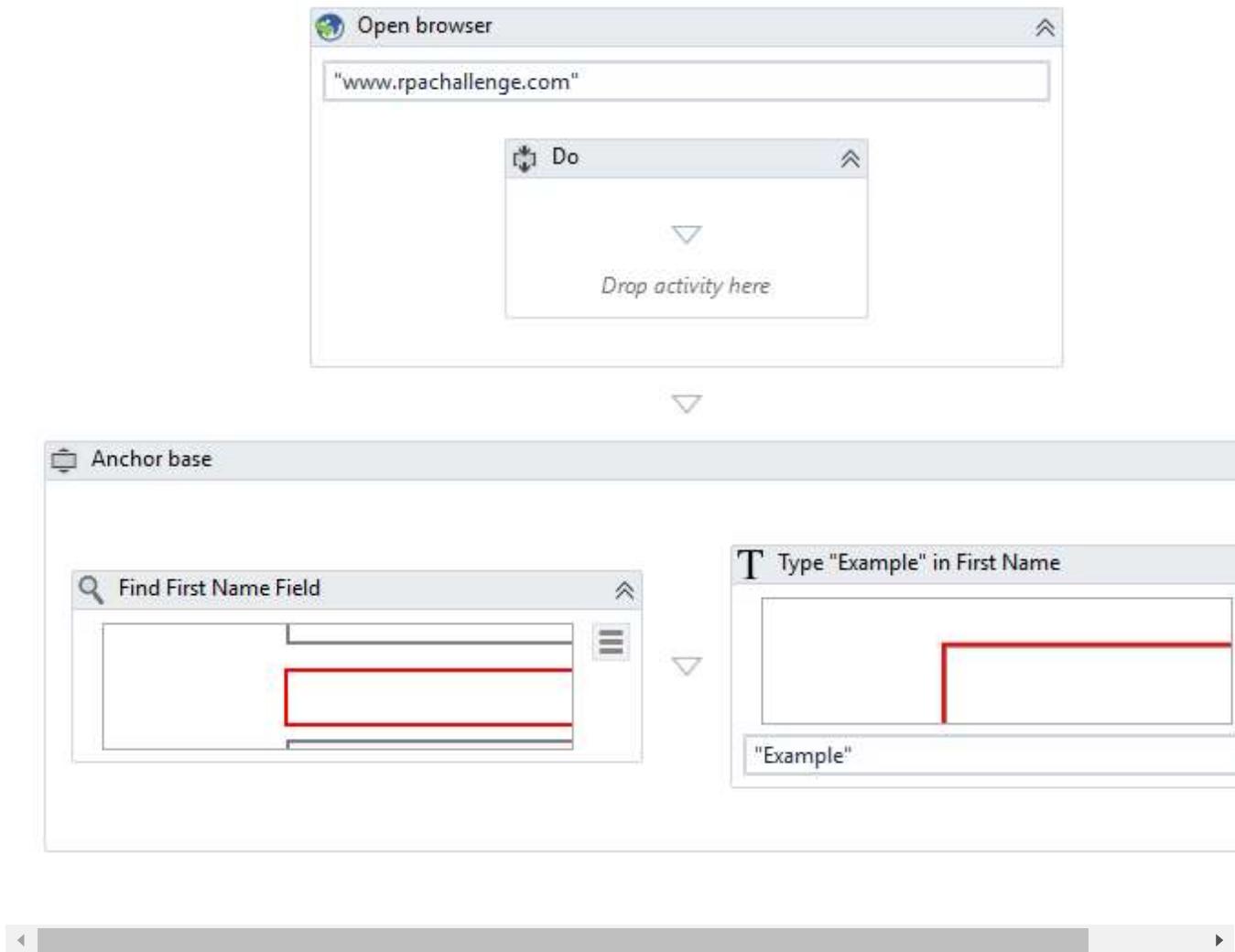
If this activity is included in **Try Catch** and the value of the **ContinueOnError** property is True, no error is caught when the project is executed.

## Misc

- **Private** - If selected, the values of variables and arguments are no longer logged at Verbose level.

# Example of using the Anchor Base activity

Anchors are used when you want to interact with an element that has an unstable selector. In this example, the workflow writes a particular text string in a field which changes its position every time the [www.rpachallenge.com](http://www.rpachallenge.com) web page loads.



The workflow contains the following activities:

- **Open Browser** - Opens www.rpachallenge.com in Internet Explorer.
- **Anchor Base** - Identifies the target field and writes the sample text:
  - Left side - The **Find Element** activity identifies the **First Name** field.
  - Right side - The **Type Into** activity writes "Example" in the **First Name** field.

[Download example](#)



Default Theme

English

# UI Automation Activities

## TABLE OF CONTENTS

### [Find Relative Element](#)

#### Properties

Example of using the Find Relative Element activity

## Find Relative Element

`UiPath.Core.Activities.FindRelative`

Searches for a UI element by using a position relative to a fixed element. This should be used when a reliable selector is not available.

## Properties

### Options

- **CursorPosition.OffsetX** - Horizontal displacement of the cursor position according to the option selected in the Position field.
- **CursorPosition.OffsetY** - Vertical displacement of the cursor position according to the option selected in the Position field.

- **CursorPosition.Position** - Describes the starting point of the cursor to which offsets from OffsetX and OffsetY properties are added. The following options are available: TopLeft, TopRight, BottomLeft, BottomRight and Center. The default option is Center.

## Output

- **RelativeElement** - The relative UI element that you are looking for. Only UIElement variables are supported.

## Misc

- **Private** - If selected, the values of variables and arguments are no longer logged at Verbose level.
- **Target.Selector** - Text property used to find a particular UI element when the activity is executed. It is actually a XML fragment specifying attributes of the GUI element you are looking for and of some of its parents.
- **Target.TimeoutMS** - Specifies the amount of time (in milliseconds) to wait for the activity to run before the SelectorNotFoundException error is thrown. The default value is 30000 milliseconds (30 seconds).
- **Target.WaitForReady** - Before performing the actions, wait for the target to become ready. The following options are available:
  - **None** - Does not wait for anything except the target UI element to exist before executing the action. For example, you can use this option if you want to retrieve just text from a web page or click a particular button, without having to wait for all UI elements to load. Note that this may have unwanted consequences if the button relies on elements which are not yet loaded, such as scripts.
  - **Interactive/Complete** - Waits all of the UI elements in the target app to exist before actually executing the action.

To assess if an application is in the Interactive or Complete state, the following tags are verified:

- **Desktop applications** - A `wm_null` message is sent to check the existence of the `<wnd>`, `<ctrl>`, `<java>`, or `<uia>` tags. If they exist, the activity is executed.
- **Web applications:**
  1. **Internet Explorer** - The `<webctrl>` tag is used to check if the **Ready** state of the HTML document is set to **Complete**. Additionally, the **Busy** state has to be set to "False".
  2. **Others** - The `<webctrl>` tag is used to check if the **Ready** state of the HTML document is **Complete**.
- **SAP applications** - First the presence of the `<wnd>` tag verified, after which a SAP specific API is used to detect if the session is busy or not.
- **Target.Element** - Use the `UIElement` variable returned by another activity. This property cannot be used alongside the `Selector` property. This field supports only `UIElement` variables.

- **Target.ClippingRegion** - Defines the clipping rectangle, in pixels, relative to the **UiElement**, in the following directions: left, top, right, bottom. It supports both positive and negative numbers.

## Common

- **DisplayName** - The display name of the activity.
- **ContinueOnError** - Specifies if the automation should continue even when the activity throws an error. This field only supports Boolean values (True, False). The default value is False. As a result, if the field is blank and an error is thrown, the execution of the project stops. If the value is set to True, the execution of the project continues regardless of any error.

 **NOTE:**

If this activity is included in **Try Catch** and the value of the **ContinueOnError** property is True, no error is caught when the project is executed.

# Example of using the Find Relative Element activity

[Here](#) you can see how the **Find Relative Element** activity is used in an example that incorporates multiple activities.



Default Theme

English

# Studio User Guide

RELEASE: 2023.4

## TABLE OF CONTENTS

[Output or Screen Scraping Methods](#)

## Output or Screen Scraping Methods

Output or screen scraping methods refer to those activities that enable you to extract data from a specified UI element or document, such as a .pdf file.

To understand which one is better for automating your business process, let's see the differences between them.

Capability Method	Speed	Accuracy	Background Execution	Extract Text Position	Extract Hidden Text	Support for Citrix
<b>FullText</b>	10/10	100%	yes	no	yes	no
<b>Native</b>	8/10	100%	no	yes	no	no
<b>OCR</b>	3/10	98%	no	yes	no	yes

**FullText** is the default method, it is fast and accurate, yet unlike the **Native** method, it cannot extract the screen coordinates of the text.

Both these methods work only with desktop applications, but the **Native** method only works with apps that are built to render text with the Graphics Device Interface (GDI).

**OCR** is not 100% accurate but can be useful to extract text that the other two methods could not, as it works with all applications including Citrix. Studio uses two OCR engines, by default: Google Tesseract and Microsoft Modi.

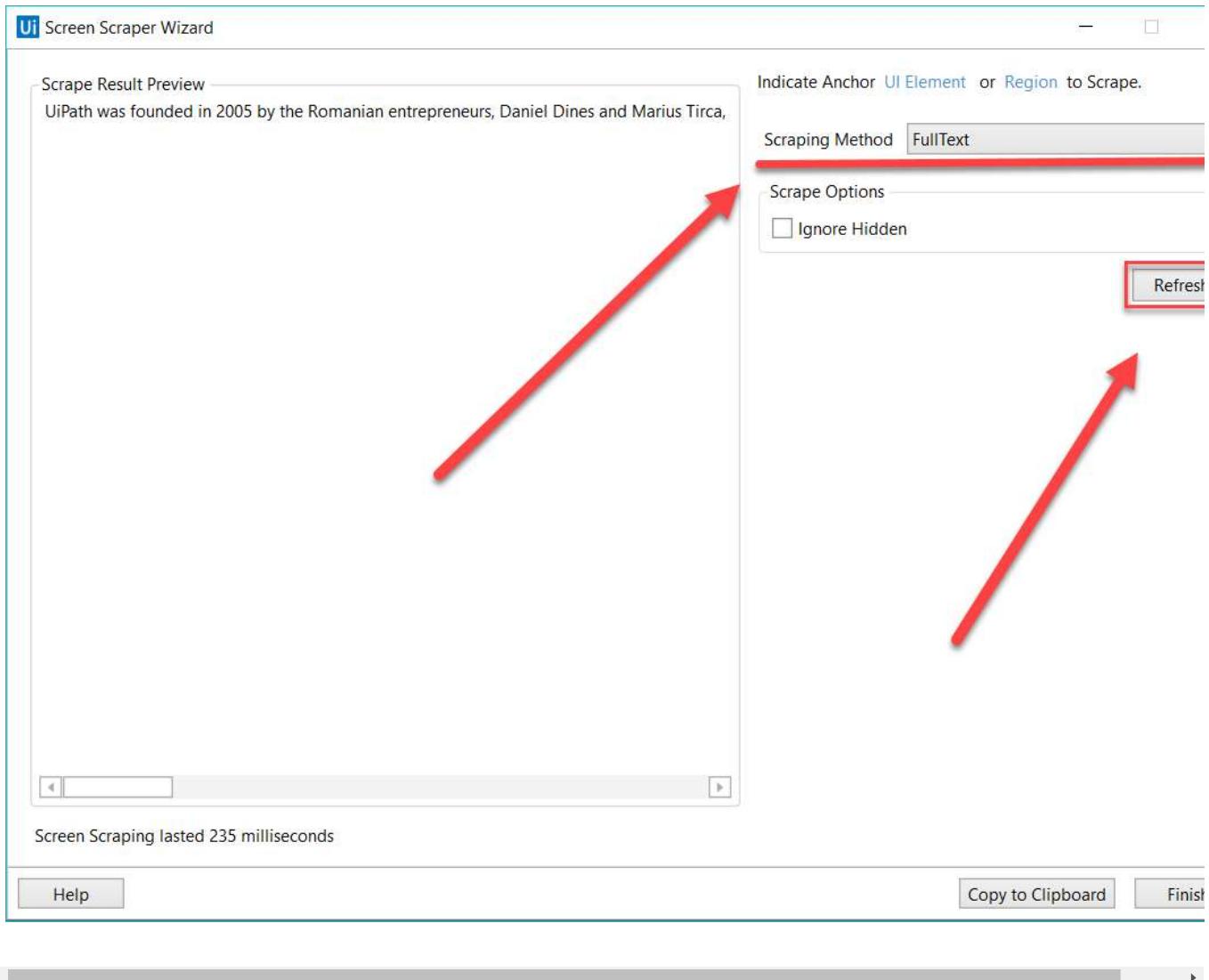
Languages can be changed for OCR engines and you can find out how to [Install OCR Languages](#) here.

Capability Method	Multiple Languages Support	Preferred Area Size	Support for Color Inversion	Set Expected Text Format	Filter Allowed Characters	Best with Microsoft Fonts
<b>Google Tesseract</b>	Can be added	Small	yes	yes	yes	no
<b>Microsoft</b>	Supported by	Large	no	no	no	yes

Capability Method	Multiple Languages Support	Preferred Area Size	Support for Color Inversion	Set Expected Text Format	Filter Allowed Characters	Best with Microsoft Fonts
MODI	default					

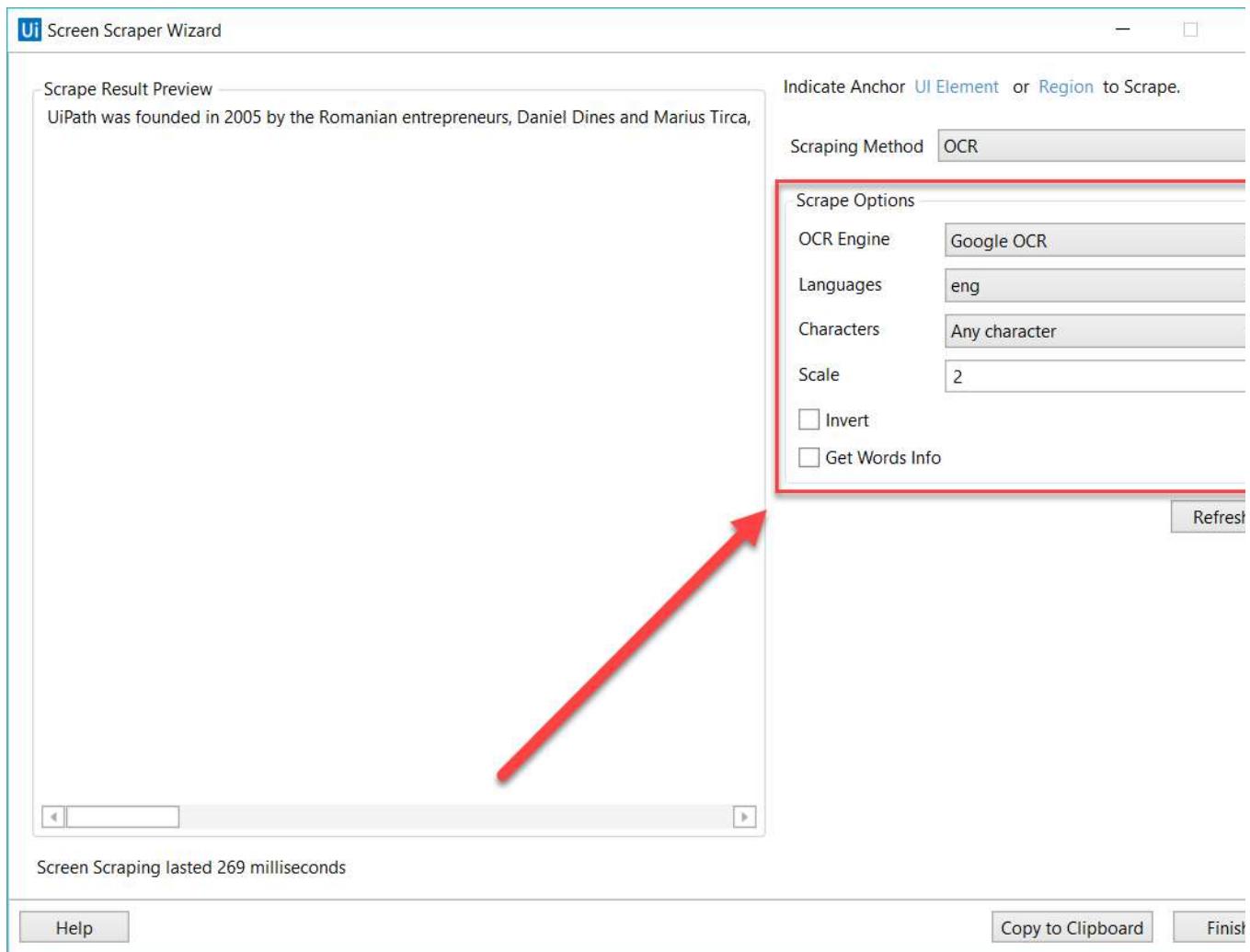
To start extracting text from various sources, click the **Screen Scraping** button, in the **Wizards** group, on the **Design** ribbon tab.

The screen scraping wizard enables you to point at a UI element and extract text from it, using one of the three output methods described above. Studio automatically chooses a screen scraping method for you, and displays it at the top of the **Screen Scraper Wizard** window.



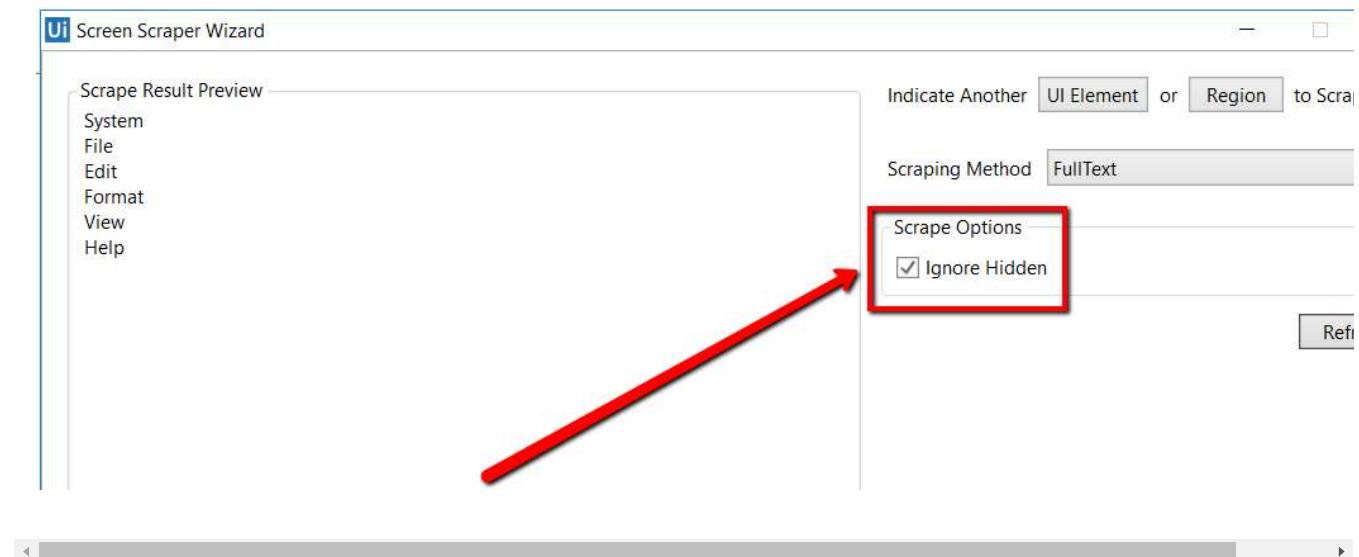
To change the method of screen scraping, select another one from the **Options** panel and then click **Refresh**.

When you are satisfied with the scraping results, click **Copy to Clipboard** and then **Finish**. The latter option copies the extracted text to the Clipboard, and it can be added to a **Generate Data Table** activity in the Designer panel. Just like [desktop recording](#), screen scraping generates a container (with the selector of the top-level window) which contains activities, and partial selectors for each activity.



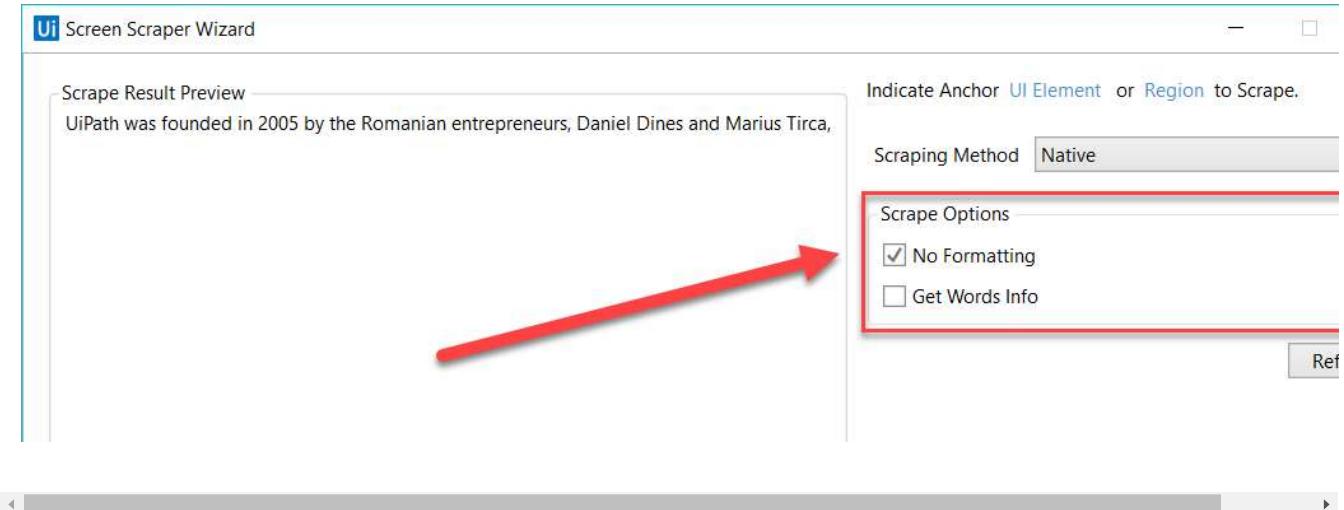
Each type of screen scraping comes with different features in the **Screen Scraper Wizard**, in the **Options** panel:

#### 1. FullText



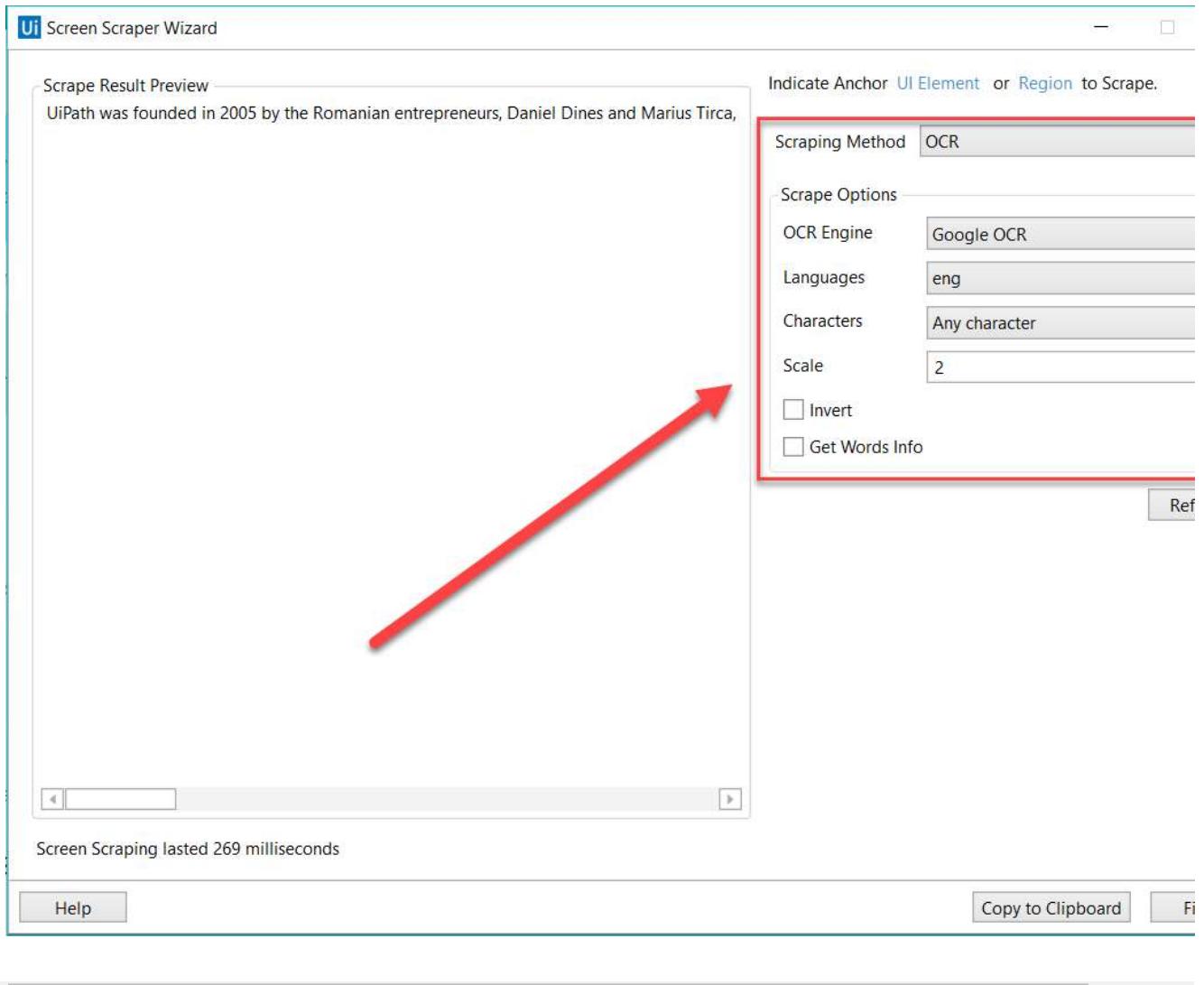
- **Ignore Hidden** – when this checkbox is selected, the hidden text from the selected UI element is not copied.

## 2. Native



- **No Formatting** – when this checkbox is selected, the copied text does not extract formatting information from the text. Otherwise, the extracted text's relative position is retained.
- **Get Words Info** – when this checkbox is selected, Studio also extracts the screen coordinates of each word. Additionally, the Custom Separators field is displayed, which enables you to specify the characters used as separators. If the field is empty, all known text separators are used.

## 3. Google OCR

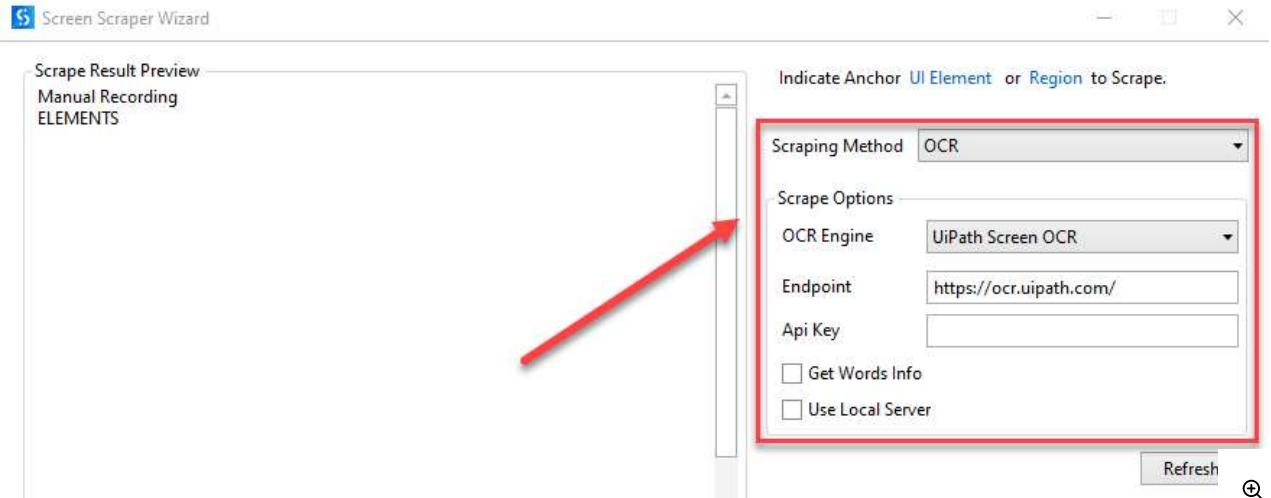


- **Languages** – only English is available by default.
- **Characters** – enables you to select which types of characters to be extracted. The following options are available: **Any character**, **Numbers only**, **Letters**, **Uppercase**, **Lowercase**, **Phone numbers**, **Currency**, **Date** and **Custom**. If you select **Custom**, two additional fields, **Allowed** and **Denied**, are displayed that enable you to create custom rules on which types of characters to scrape and which to avoid.
- **Invert** – when this checkbox is selected, the colors of the UI element are inverted before scraping. This is useful when the background is darker than the text color.
- **Scale** – the scaling factor of the selected UI element or image. The higher the number is, the more you enlarge the image. This can provide a better OCR read and it is recommended with small images.
- **Get Words Info** – gets the on-screen position of each scraped word.

**NOTE:**

In some instances of UiPath Studio, the Google Tesseract engine may have training files (about training files: [Wikipedia](#), [GitHub](#)) that do not work for certain non-English languages. Running a project with these corrupted training files may lead to an exception being thrown. To fix this issue, download the training file for the language you wish to use from [here](#) and copy it into the tessdata folder from the UiPath installation directory. To check if the training files you downloaded work, you can download this [test project](#).

#### 4. UiPath Screen OCR

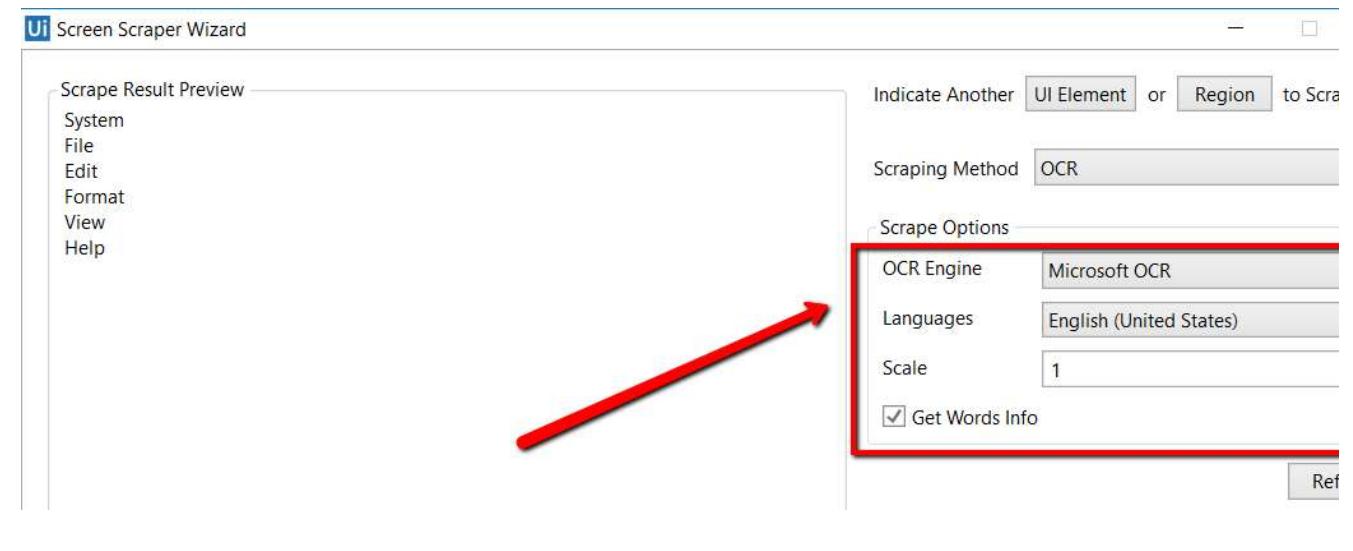


- **Endpoint** – the endpoint where the OCR model is hosted, either publicly or through an [ML Skill in AI Center](#).
- **API Key** – the endpoint API key.
- **Get Words Info** – gets the on-screen position of each scraped word.
- **Use Local Server** – select this option if you want to run the OCR locally (requires [Computer Vision Local Server Pack](#))

## 5. Microsoft OCR

### **⚠️ IMPORTANT:**

Microsoft OCR scraping engine does not support .NET 5 workflows.



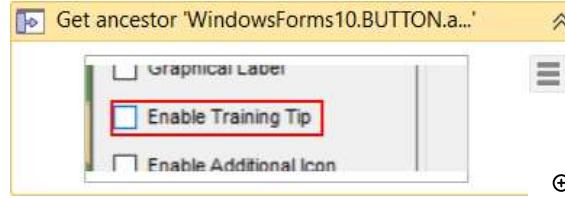
- **Languages** – enables you to change the language of the scraped text. By default, English is selected.
- **Scale** – the scaling factor of the selected UI element or image. The higher the number is, the more you enlarge the image. This can provide a better OCR read and it is recommended with small images.
- **Get Words Info** – gets the on-screen position of each scraped word.

Besides getting text out of an indicated UI element, you can also extract the value of multiple types of attributes, its exact screen position, and its ancestor.

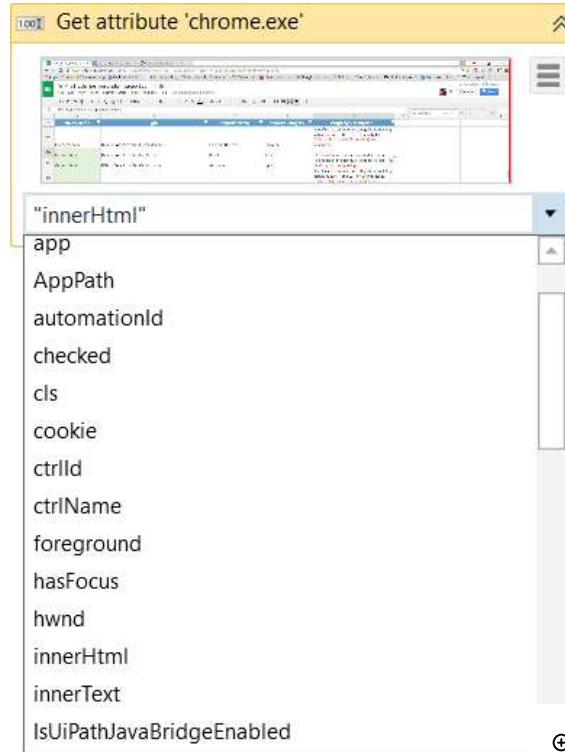
This type of information can be extracted through dedicated activities that are found in the **Activities** panel, under **UI Automation > Element > Find and UI Automation > Element > Attribute**.

These activities are:

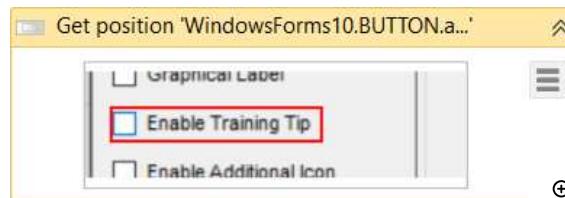
- [Get Ancestor](#) – enables you to retrieve an ancestor from a specified UI element. You can indicate at which level of the UI hierarchy to find the ancestor, and store the results in a `UiElement` variable.



- [Get Attribute](#) – retrieves the value of a specified UI element attribute. Once you indicate the UI element on screen, a drop-down list with all available attributes is displayed.



- [Get Position](#) – retrieves the bounding rectangle of the specified `UiElement`, and supports only `Rectangle` variables.



UiPath Studio also features **Relative Scraping**, a scraping method that identifies the location of the text to be retrieved relative to an anchor. You can find more about it [here](#).

You can also generate tables from unstructured data and store the information in DataTable variables, by using the **Screen Scraping Wizard**. For more information, see [Generating Tables from Unstructured Data](#).



Default Theme

English

# Studio User Guide

RELEASE: 2023.4 ▾

## TABLE OF CONTENTS

### [UI Automation](#)

[Desktop Automation](#)

[Input Methods](#)

[Selectors](#)

[Containers](#)

[Image Automation](#)

[Resolution Considerations](#)

[OCR Engines](#)

[UI Synchronization](#)

[Background Automation](#)

[How-tos](#)

[How to Execute Chrome/Edge/Firefox Automation in PiP Without Interrupting Your Work in the Main Session](#)

## UI Automation

### **IMPORTANT:**

Automation processes that use UIAutomation activities cannot run under a locked screen.

Sometimes the usual manual routine is not the optimal way for automation. Carefully explore the application's behavior and UiPath's integration/features before committing to a certain approach.

### **NOTE:**

Using the [Parallel](#) activity with UI Automation activities is not supported and will often result in unforeseen consequences.

## Desktop Automation

UI automation is best utilized when Robots and applications run on the same machine because UiPath can integrate directly with the technology behind the application to identify elements, trigger events and get data behind the scenes.

## Input Methods

There are three methods UiPath uses for triggering a **Click** or a **Type Into** activity on an application. These are displayed as properties in all activities that deal with UI automation.

### The SimulateType and SimulateClick Properties

If the **SimulateType** or **SimulateClick** properties are selected, Studio hooks into the application and triggers the event handler of an indicated UI element (button, text box). The outcome is always dependent of the target technology, such as a web browser or business application.

It is highly recommended to check the state of the target UI element prior to execution, when you enable the **SimulateType** or **SimulateClick** for the following activities:

- [Click](#)
- [Type Into](#)
- [Type Secure Text](#)

The same recommendation should be considered for the applications below, because they use the **SimulateType** or **SimulateClick** properties by default and can not be changed:

- [Check](#)
- [Select Item](#)
- [Select Multiple Items](#)
- [Set Text](#)

### The AlterIfDisabled Property

The property instructs the target activity whether or not to interact with disabled elements. Please note that this property is only taken into consideration if the **SimulateType** or **SimulateClick** are enabled. You can find the **AlterIfDisabled** property for the following activities:

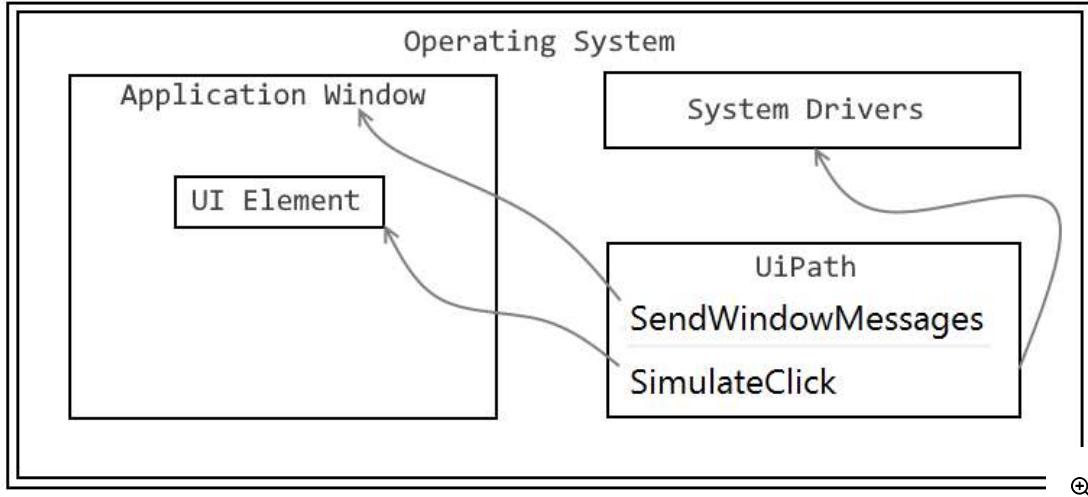
- [Click](#)
- [Type Into](#)
- [Type Secure Text](#)
- [Check](#)
- [Select Item](#)
- [Select Multiple Items](#)
- [Set Text](#)

### The SendWindowMessages Property

If the **SendWindowMessages** property is selected, Studio posts the event details to the application message loop and the application's window procedure dispatches it to the target UI element internally.

<input type="checkbox"/>
<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>
[ ] <input type="button" value="+"/>

Studio signals system drivers with hardware events if none of the above option are selected and lets the operating system dispatch the details towards the target element.



These methods should be tried in the order presented, as the **SimulateClick** and **SendWindowMessages** properties are faster and also work in the background, but they depend mostly on the technology behind the application.

Hardware events work 100% as Studio performs actions just like a human operator, such as moving the mouse pointer and clicking at a particular location. However, in this case, the application being automated needs to be visible on the screen. This can be seen as a drawback since there is the risk that the user can interfere with the automation.

## Selectors

Sometimes the automatically generated selectors propose volatile attribute values to identify elements and manual intervention is required to calibrate the selectors. A reliable selector should successfully identify the same element every time in all conditions, in development, test and production environments and no matter the usernames logged on to the applications.

Here are some tips on how to improve a selector in the [Selector Editor](#) or [UI Explorer](#):

- Replace attributes with volatile values with attributes that look steady and meaningful.
- Replace variable parts of an attribute value with wildcards (\*).
- If an attribute's value is all wildcard (e.g. name='\*') then the attribute should be removed.
- If editing attributes doesn't help, try adding more intermediary containers.
- Avoid using the idx attribute unless its value is a very small number like 1 or 2.

S	M	T	W	T	F	S
25	26	27	28	29	30	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	<b>31</b>	1	2	3	4	5

**Selector of current UI element is listed below.**

```
<html title='Google Calendar - Week of Oct 16, 2016'>
<webctrl id='dp_0_tbl' tag='TABLE'>
<webctrl id='dp_0_day_23879' tag='TD'>
```

**InitializeFromSelector: 0 ms**

Selector attributes	Value
aaname	7
class	dp-cell dp-
colName	F
css-selector	body>div>
<input checked="" type="checkbox"/> id	dp_0_day.
innertext	7
isleaf	1

In the selector above, we notice the page title has a reference to the time when the selector was recorded and also that some attributes have randomly looking IDs. Tweaking the attributes, we can come up with a better selector than UiPath recorder proposed.

Selector of current UI element is listed below.

```
<html title='Google Calendar - *'>
<webctrl aaname='S M T W T F S' tag='TABLE'>
<webctrl aaname='7' tag='TD'>
```

InitializeFromSelector: 0 ms

Selector attributes	Value
<input checked="" type="checkbox"/> aaname	7
<input type="checkbox"/> class	dp-cell dp-week
<input type="checkbox"/> colName	F
<input type="checkbox"/> css-selector	body>div>div>d
<input type="checkbox"/> id	dp_0_day_2387

## Containers

Similar to file paths, selectors can be full or partial (relative). Full selectors start with a window or an HTML identifier and have all the necessary information to find an element on the whole desktop, while partial selectors work only inside an attach/container that specifies the top-level window where the elements belong:

- [OpenBrowser](#)
- [OpenApplication](#)
- [AttachBrowser](#)
- [AttachWindow](#)

Here is a bit more info on [Full versus Partial Selectors](#).

There are several advantages to using containers with partial selectors instead of full selectors:

- Visually groups activities that work on the same application.
- Is slightly faster, not seeking for the top window every time.
- Makes it easier to manage top-level selectors in case manual updates are necessary.
- Essential when working on two instances of the same application.

## Image Automation

Image recognition is the last approach to automating applications if nothing else works to identify UI elements on the screen (like selectors or keyboard shortcuts). Because image matching requires elements to be fully visible on the screen and that all visible details are the same at runtime as during development, when resorting to image automation extra care should be taken to ensure the reliability of the process. Selecting more/less of an image than needed might lead to an image not found or a false-positive match.

## Resolution Considerations

Image matching is sensitive to environment variations like desktop theme or screen resolution. When the application runs in Citrix, the resolution should be kept greater or equal than when recording the workflows. Otherwise, small image distortions can be compensated by slightly lowering the captured image Accuracy factor. Check how the application layout adjusts itself to different resolutions to ensure visual elements proximity, especially in the case of coordinate based techniques like relative click and relative scrape.

If the automation supports different resolutions, parallel recordings can be placed inside a **PickBranch** activity and the Robot uses either match.

## OCR Engines

If OCR returns good results for the application, text automation is a good alternative to minimize the environment influence. The Google Tesseract engine works better for smaller areas, while Microsoft MODI for larger ones.

Using the MODI engine in loop automations can sometimes create memory leaks. This is why it is recommended that scraping done with MODI be invoked via a separate workflow, using the **Isolated** property.

# UI Synchronization

Unexpected behavior is likely to occur when the application is not in the state the workflow assumes it to be. The first thing to watch for is the time the application takes to respond to the Robot interactions.

The **DelayMS** property enables you to wait a while for the application to respond. However, there are situations when an application's state must be validated before proceeding with certain steps in a process. Measures may include using extra activities that wait for the desired application state before other interactions. Activities that might help include:

- [Element Exists](#), [Image Exists](#), [Text Exists](#), [OCR Text Exists](#).
- [Find Element](#), [Find Image](#), [Find Text Position](#).
- [Wait Element Vanish](#), [Wait Image Vanish](#).
- [Wait Screen Text](#) (in terminals).

# Background Automation

If an automation is intended to share the desktop with a human user, all UI interaction must be implemented in the background. This means that the automation has to work with UI element objects directly, thus allowing the application window to be hidden or minimized during the process.

- Use the [SimulateType](#), [SimulateClick](#), and [SendWindowMessagesoptions](#) for navigation and data entry via the [Click](#) and [Type Into](#) activities.
- Use the [Set Text](#), [Check](#), and [Select Item](#) activities for background data entry.
- The [Get Text](#), [Get Full Text](#), and [WebScraping](#) activities are the outputs that run in the background.
- Use the [Element Exists](#) activity to verify application state.

## How-tos

### How to Execute Chrome/Edge/Firefox Automation in PiP Without Interrupting Your Work in the Main Session

When executing web automation with Chrome/Edge/Firefox browsers in the child session of a PictureInPicture (PiP) mode and you want to continue using the browser in the main session, please know that both Chrome and Edge cannot be launched simultaneously, in different sessions, while using the same [user data directory](#).

The default folder is created by the browser, at its first launch. On Windows, the default folder location is %LOCALAPPDATA%\Google\Chrome\User Data. Your browser profiles data is kept here and every new browser profile you create has its own space: navigation history, bookmarks, and cookies.

If, as a user, for any particular reason, you choose to start the browser with your own custom data folder, you can do that by launching the browser with the following command: chrome.exe --user-data-dir=c:\foo. All your profiles data will be kept under that custom folder.

To bypass this limitation, the Modern [UseApplicationBrowser](#) and the Classic [OpenBrowser](#) activities provide an out-of-the-box solution: the **UserDataFolderPath** properties which instruct the activity on launching the browser with either its default folder, the one automatically generated by the activity, or a custom one, specified by the latter property.

Let's take a closer look at how these properties work.

#### Case 1

You don't want to know all the details about the user data folder, you simply want the automation to launch the browser in PiP and do its job, allowing you to do your own job in the main session. Then, just leave the **UserDataFolderPath** empty. Its default value will be inherited as Automatic from the Project Settings/ApplicationBrowser section: the robot will automatically create for you a temporary data folder/profile to run the automation in PiP.

#### Case 2

You want to execute the automation in PiP with the default user data folder since it contains your default profile and all your business apps login history and cookies. Set **UserDataFolderPath** to **DefaultFolder**. Please keep in mind that simply launching the browser in your working session is not be possible until the PiP automation finishes and/or the PiP browser stops.

#### Case 3

You know what you're doing and you understand both the browser limitation and the UiPath provided capabilities. Set **UserDataFolderMode** to **CustomFolder** and set **UserDataFolderPath** to a path of your choice.

For all of the above cases, we recommend [installing the extension per group policy](#). This way, the extension is available and activated for any browser profiles, regardless if they are created in default or custom user data folders.



---

Default Theme

English

# Workflow Activities

## TABLE OF CONTENTS

### [Invoke Code](#)

Description

Project compatibility

Cross-platform configuration

Windows - Legacy, Windows configuration

Properties panel

Example of Using the Invoke Code Activity

## Invoke Code

`UiPath.Core.Activities.InvokeCode`

## Description

Synchronously invokes **VB.NET** or **C#** code, optionally passing it a list of input arguments. This activity can also return out arguments to the caller workflow.

**ⓘ NOTE:**

The Invoke Code activity from **UiPath.System.Activities** pack version 21.10 is only compatible with **UiPath Studio 21.10** or newer versions.

 **ⓘ NOTE:**

The assemblies referenced by your code need to be added into the **Imports** panel in order to function. You can find more info [here](#).

 **ⓘ NOTE:**

Due to internal changes, this activity will no longer be visible in the Favorites list when upgrading to v20.4 or newer, if it was added to the Favorites list with a v19.4 or older version of **UiPath.System.Activities**.

# Project compatibility

[Windows - Legacy](#) | [Windows](#) | [Cross-platform](#)

## Cross-platform configuration

- **Code** - The code that is to be invoked. This field supports only strings and string variables.
- **Language** - A drop-down menu that specifies what language the invoked code is written in. The available options are **VBNet** and **CSharp**.
- **Arguments** - The parameters that can be passed to the code that is invoked.

## Windows - Legacy, Windows configuration

### Properties panel

#### Common

- **Continue On Error** - Specifies if the automation should continue even when the activity throws an error. This field only supports Boolean values (True, False). The default value is False. As a result, if the field is blank and an error is thrown, the execution of the project stops. If the value is set to True, the execution of the project continues regardless of any error.

 **ⓘ NOTE:**

If this activity is included in **Try Catch** and the value of the **Continue On Error** property is

True, no error is caught when the project is executed.

- **DisplayName** - The display name of the activity.

## Input

- **Arguments** - The parameters that can be passed to the code that is invoked.
- **Code** - The code that is to be invoked. This field supports only strings and String variables.
- **Language** - A drop-down menu that specifies what language the invoked code is written in. The available options are **VBNet** and **CSharp**.

## Misc

- **Private** - If selected, the values of variables and arguments are no longer logged at Verbose level.

# Example of Using the Invoke Code Activity

The **Invoke Code** activity is used for directly calling vb.net code from the **UiPath** platform. Using this activity you can specify exactly what code to be executed and what arguments to be passed to and from the activity.

This is how the automation process can be built:

1. Open Studio and create a new **Process**.
2. Drag a **Sequence** to the Workflow Designer.

- Create the following variables:

Variable Name	Variable Type	Default Value
ExampleInArgument	String	
OutValue	Int32	

3. Drag an **Assign** activity inside the sequence container.

- Add the variable ExampleInArgument in the **To** field and the expression "Example for In Arguments" in the **Value** field.

4. Add an **Invoke Code** activity below the **Assign** activity.

- Select the **Edit Code** button for introducing the desired code.

- Type the code in the **Code Editor** window. For this example we have added a simple code that writes a string on the screen. Here is the code we have used:

```
Dim TextToWrite As String
TextToWrite = "Example"
Console.WriteLine(TextToWrite)
```

- Select the **OK** button to close the **Code Editor** window.
- Select the **Edit Arguments** button and create the following arguments:

Argument Name	Argument Direction	Argument Type	Argument Value
ExampleInArgument	In	String	ExampleInArgument
ExampleOutArgument	Out	Int32	OutValue

- Select the **OK** button to close the **Invoked code arguments** window.

## 5. Place a **Write Line** activity under the **Invoke Code** activity.

- Add the expression `OutValue.ToString` in the **Text** field.

## 6. Run the process. The robot displays in the **Output** panel the code and arguments specified by you in the **Invoke Code** activity.

[Download example](#)



Default Theme

English

# List<T> Class

Reference

## Definition

Namespace: [System.Collections.Generic](#)

Assembly: System.Collections.dll

Represents a strongly typed list of objects that can be accessed by index. Provides methods to search, sort, and manipulate lists.

C#

```
public class List<T> : System.Collections.Generic.ICollection<T>,
System.Collections.Generic.IEnumerable<T>,
System.Collections.Generic.IList<T>,
System.Collections.Generic.IReadOnlyCollection<T>,
System.Collections.Generic.IReadOnlyList<T>, System.Collections.IList
```

## Type Parameters

T

The type of elements in the list.

Inheritance [Object](#) → List<T>

Derived [Microsoft.Extensions.Telemetry.Logging.LogMethodHelper](#)  
[System.Data.Services.ExpandSegmentCollection](#)  
[System.Workflow.Activities.OperationParameterInfoCollection](#)  
[System.Workflow.Activities.WorkflowRoleCollection](#)  
[System.Workflow.ComponentModel.ActivityCollection](#)  
[More...](#)

Implements [ICollection<T>](#) , [IEnumerable<T>](#) , [IList<T>](#) , [IReadOnlyCollection<T>](#) ,  
[IReadOnlyList<T>](#) , [ICollection](#) , [IEnumerable](#) , [IList](#)

## Examples

The following example demonstrates how to add, remove, and insert a simple business object in a `List<T>`.

C#

```
using System;
using System.Collections.Generic;
// Simple business object. A PartId is used to identify the type of part
// but the part name can change.
public class Part : IEquatable<Part>
{
    public string PartName { get; set; }

    public int PartId { get; set; }

    public override string ToString()
    {
        return "ID: " + PartId + " Name: " + PartName;
    }
    public override bool Equals(object obj)
    {
        if (obj == null) return false;
        Part objAsPart = obj as Part;
        if (objAsPart == null) return false;
        else return Equals(objAsPart);
    }
    public override int GetHashCode()
    {
        return PartId;
    }
    public bool Equals(Part other)
    {
        if (other == null) return false;
        return (this.PartId.Equals(other.PartId));
    }
    // Should also override == and != operators.
}
public class Example
{
    public static void Main()
    {
        // Create a list of parts.
        List<Part> parts = new List<Part>();

        // Add parts to the list.
        parts.Add(new Part() { PartName = "crank arm", PartId = 1234 });
        parts.Add(new Part() { PartName = "chain ring", PartId = 1334 });
        parts.Add(new Part() { PartName = "regular seat", PartId = 1434 });
        parts.Add(new Part() { PartName = "banana seat", PartId = 1444 });
        parts.Add(new Part() { PartName = "cassette", PartId = 1534 });
        parts.Add(new Part() { PartName = "shift lever", PartId = 1634 });

        // Write out the parts in the list. This will call the overridden
    }
}
```

```
ToString method
// in the Part class.
Console.WriteLine();
foreach (Part aPart in parts)
{
    Console.WriteLine(aPart);
}

// Check the list for part #1734. This calls the IEquatable.Equals
method
// of the Part class, which checks the PartId for equality.
Console.WriteLine("\nContains(\"1734\"): {0}",
parts.Contains(new Part { PartId = 1734, PartName = "" }));

// Insert a new item at position 2.
Console.WriteLine("\nInsert(2, \"1834\")");
parts.Insert(2, new Part() { PartName = "brake lever", PartId =
1834 });

//Console.WriteLine();
foreach (Part aPart in parts)
{
    Console.WriteLine(aPart);
}

Console.WriteLine("\nParts[3]: {0}", parts[3]);

Console.WriteLine("\nRemove(\"1534\")");

// This will remove part 1534 even though the PartName is different,
// because the Equals method only checks PartId for equality.
parts.Remove(new Part() { PartId = 1534, PartName = "cogs" });

Console.WriteLine();
foreach (Part aPart in parts)
{
    Console.WriteLine(aPart);
}
Console.WriteLine("\nRemoveAt(3)");
// This will remove the part at index 3.
parts.RemoveAt(3);

Console.WriteLine();
foreach (Part aPart in parts)
{
    Console.WriteLine(aPart);
}

/*
ID: 1234  Name: crank arm
ID: 1334  Name: chain ring
ID: 1434  Name: regular seat
ID: 1444  Name: banana seat
ID: 1534  Name: cassette
```

```
ID: 1634  Name: shift lever

Contains("1734"): False

Insert(2, "1834")
ID: 1234  Name: crank arm
ID: 1334  Name: chain ring
ID: 1834  Name: brake lever
ID: 1434  Name: regular seat
ID: 1444  Name: banana seat
ID: 1534  Name: cassette
ID: 1634  Name: shift lever

Parts[3]: ID: 1434  Name: regular seat

Remove("1534")

ID: 1234  Name: crank arm
ID: 1334  Name: chain ring
ID: 1834  Name: brake lever
ID: 1434  Name: regular seat
ID: 1444  Name: banana seat
ID: 1634  Name: shift lever

RemoveAt(3)

ID: 1234  Name: crank arm
ID: 1334  Name: chain ring
ID: 1834  Name: brake lever
ID: 1444  Name: banana seat
ID: 1634  Name: shift lever

*/
}
```

The following example demonstrates several properties and methods of the `List<T>` generic class of type string. (For an example of a `List<T>` of complex types, see the [Contains](#) method.)

The parameterless constructor is used to create a list of strings with the default capacity. The [Capacity](#) property is displayed and then the [Add](#) method is used to add several items. The items are listed, and the [Capacity](#) property is displayed again, along with the [Count](#) property, to show that the capacity has been increased as needed.

The [Contains](#) method is used to test for the presence of an item in the list, the [Insert](#) method is used to insert a new item in the middle of the list, and the contents of the list are displayed again.

The default `Item[]` property (the indexer in C#) is used to retrieve an item, the `Remove` method is used to remove the first instance of the duplicate item added earlier, and the contents are displayed again. The `Remove` method always removes the first instance it encounters.

The `TrimExcess` method is used to reduce the capacity to match the count, and the `Capacity` and `Count` properties are displayed. If the unused capacity had been less than 10 percent of total capacity, the list would not have been resized.

Finally, the `Clear` method is used to remove all items from the list, and the `Capacity` and `Count` properties are displayed.

C#

```
List<string> dinosaurs = new List<string>();

Console.WriteLine("\nCapacity: {0}", dinosaurs.Capacity);

dinosaurs.Add("Tyrannosaurus");
dinosaurs.Add("Amargasaurus");
dinosaurs.Add("Mamenchisaurus");
dinosaurs.Add("Deinonychus");
dinosaurs.Add("Compsognathus");
Console.WriteLine();
foreach(string dinosaur in dinosaurs)
{
    Console.WriteLine(dinosaur);
}

Console.WriteLine("\nCapacity: {0}", dinosaurs.Capacity);
Console.WriteLine("Count: {0}", dinosaurs.Count);

Console.WriteLine("\nContains(\"Deinonychus\"): {0}",
    dinosaurs.Contains("Deinonychus"));

Console.WriteLine("\nInsert(2, \"Compsognathus\")");
dinosaurs.Insert(2, "Compsognathus");

Console.WriteLine();
foreach(string dinosaur in dinosaurs)
{
    Console.WriteLine(dinosaur);
}

// Shows accessing the list using the Item property.
Console.WriteLine("\ndinosaurs[3]: {0}", dinosaurs[3]);

Console.WriteLine("\nRemove(\"Compsognathus\")");
dinosaurs.Remove("Compsognathus");

Console.WriteLine();
```

```
foreach(string dinosaur in dinosaurs)
{
    Console.WriteLine(dinosaur);
}

dinosaurs.TrimExcess();
Console.WriteLine("\nTrimExcess()");
Console.WriteLine("Capacity: {0}", dinosaurs.Capacity);
Console.WriteLine("Count: {0}", dinosaurs.Count);

dinosaurs.Clear();
Console.WriteLine("\nClear()");
Console.WriteLine("Capacity: {0}", dinosaurs.Capacity);
Console.WriteLine("Count: {0}", dinosaurs.Count);
```

*/\* This code example produces the following output:*

**Capacity: 0**

Tyrannosaurus  
Amargasaurus  
Mamenchisaurus  
Deinonychus  
Compsognathus

**Capacity: 8**

**Count: 5**

**Contains("Deinonychus"):** True

**Insert(2, "Compsognathus")**

Tyrannosaurus  
Amargasaurus  
Compsognathus  
Mamenchisaurus  
Deinonychus  
Compsognathus

**dinosaurs[3]: Mamenchisaurus**

**Remove("Compsognathus")**

Tyrannosaurus  
Amargasaurus  
Mamenchisaurus  
Deinonychus  
Compsognathus

**TrimExcess()**

**Capacity: 5**

**Count: 5**

**Clear()**

**Capacity: 5**

```
Count: 0
*/
```

## Remarks

The [List<T>](#) class is the generic equivalent of the [ArrayList](#) class. It implements the [IList<T>](#) generic interface by using an array whose size is dynamically increased as required.

You can add items to a [List<T>](#) by using the [Add](#) or [AddRange](#) methods.

The [List<T>](#) class uses both an equality comparer and an ordering comparer.

- Methods such as [Contains](#), [IndexOf](#), [LastIndexOf](#), and [Remove](#) use an equality comparer for the list elements. The default equality comparer for type  $T$  is determined as follows. If type  $T$  implements the [IEquatable<T>](#) generic interface, then the equality comparer is the [Equals\(T\)](#) method of that interface; otherwise, the default equality comparer is [Object.Equals\(Object\)](#).
- Methods such as [BinarySearch](#) and [Sort](#) use an ordering comparer for the list elements. The default comparer for type  $T$  is determined as follows. If type  $T$  implements the [IComparable<T>](#) generic interface, then the default comparer is the [CompareTo\(T\)](#) method of that interface; otherwise, if type  $T$  implements the nongeneric [IComparable](#) interface, then the default comparer is the [CompareTo\(Object\)](#) method of that interface. If type  $T$  implements neither interface, then there is no default comparer, and a comparer or comparison delegate must be provided explicitly.

The [List<T>](#) is not guaranteed to be sorted. You must sort the [List<T>](#) before performing operations (such as [BinarySearch](#)) that require the [List<T>](#) to be sorted.

Elements in this collection can be accessed using an integer index. Indexes in this collection are zero-based.

**.NET Framework only:** For very large [List<T>](#) objects, you can increase the maximum capacity to 2 billion elements on a 64-bit system by setting the `enabled` attribute of the [`<gcAllowVeryLargeObjects>`](#) configuration element to `true` in the run-time environment.

[List<T>](#) accepts `null` as a valid value for reference types and allows duplicate elements.

For an immutable version of the [List<T>](#) class, see [ImmutableList<T>](#).

## Performance Considerations

In deciding whether to use the [List<T>](#) or [ArrayList](#) class, both of which have similar functionality, remember that the [List<T>](#) class performs better in most cases and is type safe. If a reference type is used for type  $T$  of the [List<T>](#) class, the behavior of the two classes is identical. However, if a value type is used for type  $T$ , you need to consider implementation and boxing issues.

If a value type is used for type  $T$ , the compiler generates an implementation of the [List<T>](#) class specifically for that value type. That means a list element of a [List<T>](#) object does not have to be boxed before the element can be used, and after about 500 list elements are created, the memory saved by not boxing list elements is greater than the memory used to generate the class implementation.

Make certain the value type used for type  $T$  implements the [IEquatable<T>](#) generic interface. If not, methods such as [Contains](#) must call the [Object.Equals\(Object\)](#) method, which boxes the affected list element. If the value type implements the [IComparable](#) interface and you own the source code, also implement the [IComparable<T>](#) generic interface to prevent the [BinarySearch](#) and [Sort](#) methods from boxing list elements. If you do not own the source code, pass an [IComparer<T>](#) object to the [BinarySearch](#) and [Sort](#) methods

It is to your advantage to use the type-specific implementation of the [List<T>](#) class instead of using the [ArrayList](#) class or writing a strongly typed wrapper collection yourself. That's because your implementation must do what the .NET Framework does for you already, and the common language runtime can share Microsoft intermediate language code and metadata, which your implementation cannot.

## F# Considerations

The [List<T>](#) class is used infrequently in F# code. Instead, [Lists](#), which are immutable, singly-linked lists, are typically preferred. An F# [List](#) provides an ordered, immutable series of values, and is supported for use in functional-style development. When used from F#, the [List<T>](#) class is typically referred to by the [ResizeArray<'T>](#) type abbreviation to avoid naming conflicts with F# Lists.

## Constructors

[List<T>\(\)](#)

Initializes a new instance of the [List<T>](#) class that is empty and has the default initial capacity.

<a href="#">List&lt;T&gt;(IEnumerable&lt;T&gt;)</a>	Initializes a new instance of the <a href="#">List&lt;T&gt;</a> class that contains elements copied from the specified collection and has sufficient capacity to accommodate the number of elements copied.
<a href="#">List&lt;T&gt;(Int32)</a>	Initializes a new instance of the <a href="#">List&lt;T&gt;</a> class that is empty and has the specified initial capacity.

## Properties

<a href="#">Capacity</a>	Gets or sets the total number of elements the internal data structure can hold without resizing.
<a href="#">Count</a>	Gets the number of elements contained in the <a href="#">List&lt;T&gt;</a> .
<a href="#">Item[Int32]</a>	Gets or sets the element at the specified index.

## Methods

<a href="#">Add(T)</a>	Adds an object to the end of the <a href="#">List&lt;T&gt;</a> .
<a href="#">AddRange(IEnumerable&lt;T&gt;)</a>	Adds the elements of the specified collection to the end of the <a href="#">List&lt;T&gt;</a> .
<a href="#">AsReadOnly()</a>	Returns a read-only <a href="#">ReadOnlyCollection&lt;T&gt;</a> wrapper for the current collection.
<a href="#">BinarySearch(Int32, Int32, T, IComparer&lt;T&gt;)</a>	Searches a range of elements in the sorted <a href="#">List&lt;T&gt;</a> for an element using the specified comparer and returns the zero-based index of the element.
<a href="#">BinarySearch(T)</a>	Searches the entire sorted <a href="#">List&lt;T&gt;</a> for an element using the default comparer and returns the zero-based index of the element.
<a href="#">BinarySearch(T, IComparer&lt;T&gt;)</a>	Searches the entire sorted <a href="#">List&lt;T&gt;</a> for an element using the specified comparer and returns the zero-based index of the element.
<a href="#">Clear()</a>	Removes all elements from the <a href="#">List&lt;T&gt;</a> .
<a href="#">Contains(T)</a>	Determines whether an element is in the <a href="#">List&lt;T&gt;</a> .
<a href="#">ConvertAll&lt;TOutput&gt;(Converter&lt;T,TOutput&gt;)</a>	Converts the elements in the current <a href="#">List&lt;T&gt;</a> to another type, and returns a list containing the converted elements.
<a href="#">CopyTo(Int32, T[], Int32, Int32)</a>	Copies a range of elements from the <a href="#">List&lt;T&gt;</a> to a compatible one-dimensional array, starting at the specified index of the

target array.

<a href="#">CopyTo(T[])</a>	Copies the entire <code>List&lt;T&gt;</code> to a compatible one-dimensional array, starting at the beginning of the target array.
<a href="#">CopyTo(T[], Int32)</a>	Copies the entire <code>List&lt;T&gt;</code> to a compatible one-dimensional array, starting at the specified index of the target array.
<a href="#">EnsureCapacity(Int32)</a>	Ensures that the capacity of this list is at least the specified <code>capacity</code> . If the current capacity is less than <code>capacity</code> , it is successively increased to twice the current capacity until it is at least the specified <code>capacity</code> .
<a href="#">Equals(Object)</a>	Determines whether the specified object is equal to the current object. (Inherited from <a href="#">Object</a> )
<a href="#">Exists(Predicate&lt;T&gt;)</a>	Determines whether the <code>List&lt;T&gt;</code> contains elements that match the conditions defined by the specified predicate.
<a href="#">Find(Predicate&lt;T&gt;)</a>	Searches for an element that matches the conditions defined by the specified predicate, and returns the first occurrence within the entire <code>List&lt;T&gt;</code> .
<a href="#">FindAll(Predicate&lt;T&gt;)</a>	Retrieves all the elements that match the conditions defined by the specified predicate.
<a href="#">FindIndex(Int32, Int32, Predicate&lt;T&gt;)</a>	Searches for an element that matches the conditions defined by the specified predicate, and returns the zero-based index of the first occurrence within the range of elements in the <code>List&lt;T&gt;</code> that starts at the specified index and contains the specified number of elements.
<a href="#">FindIndex(Int32, Predicate&lt;T&gt;)</a>	Searches for an element that matches the conditions defined by the specified predicate, and returns the zero-based index of the first occurrence within the range of elements in the <code>List&lt;T&gt;</code> that extends from the specified index to the last element.
<a href="#">FindIndex(Predicate&lt;T&gt;)</a>	Searches for an element that matches the conditions defined by the specified predicate, and returns the zero-based index of the first occurrence within the entire <code>List&lt;T&gt;</code> .
<a href="#">FindLast(Predicate&lt;T&gt;)</a>	Searches for an element that matches the conditions defined by the specified predicate, and returns the last occurrence within the entire <code>List&lt;T&gt;</code> .
<a href="#">FindLastIndex(Int32, Int32, Predicate&lt;T&gt;)</a>	Searches for an element that matches the conditions defined by the specified predicate, and returns the zero-based index of the last occurrence within the range of elements in the <code>List&lt;T&gt;</code> that contains the specified number of elements and ends at the specified index.

<a href="#">FindLastIndex(Int32, Predicate&lt;T&gt;)</a>	Searches for an element that matches the conditions defined by the specified predicate, and returns the zero-based index of the last occurrence within the range of elements in the <a href="#">List&lt;T&gt;</a> that extends from the first element to the specified index.
<a href="#">FindLastIndex(Predicate&lt;T&gt;)</a>	Searches for an element that matches the conditions defined by the specified predicate, and returns the zero-based index of the last occurrence within the entire <a href="#">List&lt;T&gt;</a> .
<a href="#">ForEach(Action&lt;T&gt;)</a>	Performs the specified action on each element of the <a href="#">List&lt;T&gt;</a> .
<a href="#">GetEnumerator()</a>	Returns an enumerator that iterates through the <a href="#">List&lt;T&gt;</a> .
<a href="#">GetHashCode()</a>	Serves as the default hash function. (Inherited from <a href="#">Object</a> )
<a href="#">GetRange(Int32, Int32)</a>	Creates a shallow copy of a range of elements in the source <a href="#">List&lt;T&gt;</a> .
<a href="#">GetType()</a>	Gets the <a href="#">Type</a> of the current instance. (Inherited from <a href="#">Object</a> )
<a href="#">IndexOf(T)</a>	Searches for the specified object and returns the zero-based index of the first occurrence within the entire <a href="#">List&lt;T&gt;</a> .
<a href="#">IndexOf(T, Int32)</a>	Searches for the specified object and returns the zero-based index of the first occurrence within the range of elements in the <a href="#">List&lt;T&gt;</a> that extends from the specified index to the last element.
<a href="#">IndexOf(T, Int32, Int32)</a>	Searches for the specified object and returns the zero-based index of the first occurrence within the range of elements in the <a href="#">List&lt;T&gt;</a> that starts at the specified index and contains the specified number of elements.
<a href="#">Insert(Int32, T)</a>	Inserts an element into the <a href="#">List&lt;T&gt;</a> at the specified index.
<a href="#">InsertRange(Int32, IEnumerable&lt;T&gt;)</a>	Inserts the elements of a collection into the <a href="#">List&lt;T&gt;</a> at the specified index.
<a href="#">LastIndexOf(T)</a>	Searches for the specified object and returns the zero-based index of the last occurrence within the entire <a href="#">List&lt;T&gt;</a> .
<a href="#">LastIndexOf(T, Int32)</a>	Searches for the specified object and returns the zero-based index of the last occurrence within the range of elements in the <a href="#">List&lt;T&gt;</a> that extends from the first element to the specified index.
<a href="#">LastIndexOf(T, Int32, Int32)</a>	Searches for the specified object and returns the zero-based index of the last occurrence within the range of elements in the

	List<T> that contains the specified number of elements and ends at the specified index.
<a href="#">MemberwiseClone()</a>	Creates a shallow copy of the current <a href="#">Object</a> . (Inherited from <a href="#">Object</a> )
<a href="#">Remove(T)</a>	Removes the first occurrence of a specific object from the List<T>.
<a href="#">RemoveAll(Predicate&lt;T&gt;)</a>	Removes all the elements that match the conditions defined by the specified predicate.
<a href="#">RemoveAt(Int32)</a>	Removes the element at the specified index of the List<T>.
<a href="#">RemoveRange(Int32, Int32)</a>	Removes a range of elements from the List<T>.
<a href="#">Reverse()</a>	Reverses the order of the elements in the entire List<T>.
<a href="#">Reverse(Int32, Int32)</a>	Reverses the order of the elements in the specified range.
<a href="#">Sort()</a>	Sorts the elements in the entire List<T> using the default comparer.
<a href="#">Sort(Comparison&lt;T&gt;)</a>	Sorts the elements in the entire List<T> using the specified Comparison<T>.
<a href="#">Sort(IComparer&lt;T&gt;)</a>	Sorts the elements in the entire List<T> using the specified comparer.
<a href="#">Sort(Int32, Int32, IComparer&lt;T&gt;)</a>	Sorts the elements in a range of elements in List<T> using the specified comparer.
<a href="#">ToArray()</a>	Copies the elements of the List<T> to a new array.
<a href="#">ToString()</a>	Returns a string that represents the current object. (Inherited from <a href="#">Object</a> )
<a href="#">TrimExcess()</a>	Sets the capacity to the actual number of elements in the List<T>, if that number is less than a threshold value.
<a href="#">TrueForAll(Predicate&lt;T&gt;)</a>	Determines whether every element in the List<T> matches the conditions defined by the specified predicate.

## Explicit Interface Implementations

<a href="#">ICollection.CopyTo(Array, Int32)</a>	Copies the elements of the <a href="#">ICollection</a> to an <a href="#">Array</a> , starting at a particular <a href="#">Array</a> index.
<a href="#">ICollection.IsSynchronized</a>	Gets a value indicating whether access to the <a href="#">ICollection</a> is synchronized (thread safe).

<a href="#">ICollection.SyncRoot</a>	Gets an object that can be used to synchronize access to the <a href="#">ICollection</a> .
<a href="#">ICollection&lt;T&gt;.IsReadOnly</a>	Gets a value indicating whether the <a href="#">ICollection&lt;T&gt;</a> is read-only.
<a href="#">IEnumerable.GetEnumerator()</a>	Returns an enumerator that iterates through a collection.
<a href="#">IEnumerable&lt;T&gt;.Get Enumerator()</a>	Returns an enumerator that iterates through a collection.
<a href="#">IList.Add(Object)</a>	Adds an item to the <a href="#">IList</a> .
<a href="#">IList.Contains(Object)</a>	Determines whether the <a href="#">IList</a> contains a specific value.
<a href="#">IList.IndexOf(Object)</a>	Determines the index of a specific item in the <a href="#">IList</a> .
<a href="#">IList.Insert(Int32, Object)</a>	Inserts an item to the <a href="#">IList</a> at the specified index.
<a href="#">IList.IsFixedSize</a>	Gets a value indicating whether the <a href="#">IList</a> has a fixed size.
<a href="#">IList.IsReadOnly</a>	Gets a value indicating whether the <a href="#">IList</a> is read-only.
<a href="#">IList.Item[Int32]</a>	Gets or sets the element at the specified index.
<a href="#">IList.Remove(Object)</a>	Removes the first occurrence of a specific object from the <a href="#">IList</a> .

## Extension Methods

<a href="#">AsReadOnly&lt;T&gt;(IList&lt;T&gt;)</a>	Returns a read-only <a href="#">ReadOnlyCollection&lt;T&gt;</a> wrapper for the specified list.
<a href="#">ToImmutableArray&lt;TSource&gt;(IEnumerable&lt;TSource&gt;)</a>	Creates an immutable array from the specified collection.
<a href="#">ToImmutableDictionary&lt;TSource,TKey&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,TKey&gt;)</a>	Constructs an immutable dictionary from an existing collection of elements, applying a transformation function to the source keys.
<a href="#">ToImmutableDictionary&lt;TSource,TKey&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,TKey&gt;, IEqualityComparer&lt;TKey&gt;)</a>	Constructs an immutable dictionary based on some transformation of a sequence.
<a href="#">ToImmutableDictionary&lt;TSource,TKey,TValue&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,TKey&gt;, Func&lt;TSource,TValue&gt;)</a>	Enumerates and transforms a sequence, and produces an immutable dictionary of its contents.
<a href="#">ToImmutableDictionary&lt;TSource,TKey,TValue&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,TKey&gt;, Func&lt;TSource,TValue&gt;, IEqualityComparer&lt;TKey&gt;)</a>	Enumerates and transforms a sequence, and produces an immutable dictionary of its contents by using the specified key comparer.

ToImmutableDictionary<TSource,TKey,TValue> (IEnumerable<TSource>, Func<TSource,TKey>, Func<TSource,TValue>, IEqualityComparer<TKey>, IEqualityComparer<TValue>)	Enumerates and transforms a sequence, and produces an immutable dictionary of its contents by using the specified key and value comparers.
ToImmutableHashSet<TSource> (IEnumerable<TSource>)	Enumerates a sequence and produces an immutable hash set of its contents.
ToImmutableHashSet<TSource> (IEnumerable<TSource>, IEqualityComparer<TSource>)	Enumerates a sequence, produces an immutable hash set of its contents, and uses the specified equality comparer for the set type.
ToImmutableList<TSource> (IEnumerable<TSource>)	Enumerates a sequence and produces an immutable list of its contents.
ToImmutableSortedDictionary<TSource,TKey,TValue> (IEnumerable<TSource>, Func<TSource,TKey>, Func<TSource,TValue>)	Enumerates and transforms a sequence, and produces an immutable sorted dictionary of its contents.
ToImmutableSortedDictionary<TSource,TKey,TValue> (IEnumerable<TSource>, Func<TSource,TKey>, Func<TSource,TValue>, IComparer<TKey>)	Enumerates and transforms a sequence, and produces an immutable sorted dictionary of its contents by using the specified key comparer.
ToImmutableSortedDictionary<TSource,TKey,TValue> (IEnumerable<TSource>, Func<TSource,TKey>, Func<TSource,TValue>, IComparer<TKey>, IEqualityComparer<TValue>)	Enumerates and transforms a sequence, and produces an immutable sorted dictionary of its contents by using the specified key and value comparers.
ToImmutableSortedSet<TSource> (IEnumerable<TSource>)	Enumerates a sequence and produces an immutable sorted set of its contents.
ToImmutableSortedSet<TSource> (IEnumerable<TSource>, IComparer<TSource>)	Enumerates a sequence, produces an immutable sorted set of its contents, and uses the specified comparer.
CopyToDataTable<T>(IEnumerable<T>)	Returns a <a href="#">DataTable</a> that contains copies of the <a href="#">DataRow</a> objects, given an input <a href="#">IEnumerable&lt;T&gt;</a> object where the generic parameter T is <a href="#">DataRow</a> .
CopyToDataTable<T>(IEnumerable<T>, DataTable, LoadOption)	Copies <a href="#">DataRow</a> objects to the specified <a href="#">DataTable</a> , given an input <a href="#">IEnumerable&lt;T&gt;</a> object where the generic parameter T is <a href="#">DataRow</a> .
CopyToDataTable<T>(IEnumerable<T>, DataTable, LoadOption, FillErrorEventHandler)	Copies <a href="#">DataRow</a> objects to the specified <a href="#">DataTable</a> , given an input <a href="#">IEnumerable&lt;T&gt;</a>

	object where the generic parameter <code>T</code> is <a href="#">DataRow</a> .
<a href="#">Aggregate&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,TSource,TSource&gt;)</a>	Applies an accumulator function over a sequence.
<a href="#">Aggregate&lt;TSource,TAccumulate&gt;(IEnumerable&lt;TSource&gt;, TAccumulate, Func&lt;TAccumulate,TSource,TAccumulate&gt;)</a>	Applies an accumulator function over a sequence. The specified seed value is used as the initial accumulator value.
<a href="#">Aggregate&lt;TSource,TAccumulate,TResult&gt;(IEnumerable&lt;TSource&gt;, TAccumulate, Func&lt;TAccumulate,TSource,TAccumulate&gt;, Func&lt;TAccumulate,TResult&gt;)</a>	Applies an accumulator function over a sequence. The specified seed value is used as the initial accumulator value, and the specified function is used to select the result value.
<a href="#">All&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,Boolean&gt;)</a>	Determines whether all elements of a sequence satisfy a condition.
<a href="#">Any&lt;TSource&gt;(IEnumerable&lt;TSource&gt;)</a>	Determines whether a sequence contains any elements.
<a href="#">Any&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,Boolean&gt;)</a>	Determines whether any element of a sequence satisfies a condition.
<a href="#">Append&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, TSource)</a>	Appends a value to the end of the sequence.
<a href="#">AsEnumerable&lt;TSource&gt;(IEnumerable&lt;TSource&gt;)</a>	Returns the input typed as <a href="#">IEnumerable&lt;T&gt;</a> .
<a href="#">Average&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,Decimal&gt;)</a>	Computes the average of a sequence of <a href="#">Decimal</a> values that are obtained by invoking a transform function on each element of the input sequence.
<a href="#">Average&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,Double&gt;)</a>	Computes the average of a sequence of <a href="#">Double</a> values that are obtained by invoking a transform function on each element of the input sequence.
<a href="#">Average&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,Int32&gt;)</a>	Computes the average of a sequence of <a href="#">Int32</a> values that are obtained by invoking a transform function on each element of the input sequence.
<a href="#">Average&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,Int64&gt;)</a>	Computes the average of a sequence of <a href="#">Int64</a> values that are obtained by invoking a transform function on each element of the input sequence.

Average<TSource>(IEnumerable<TSource>, Func<TSource, Nullable<Decimal>>)	Computes the average of a sequence of nullable <b>Decimal</b> values that are obtained by invoking a transform function on each element of the input sequence.
Average<TSource>(IEnumerable<TSource>, Func<TSource, Nullable<Double>>)	Computes the average of a sequence of nullable <b>Double</b> values that are obtained by invoking a transform function on each element of the input sequence.
Average<TSource>(IEnumerable<TSource>, Func<TSource, Nullable<Int32>>)	Computes the average of a sequence of nullable <b>Int32</b> values that are obtained by invoking a transform function on each element of the input sequence.
Average<TSource>(IEnumerable<TSource>, Func<TSource, Nullable<Int64>>)	Computes the average of a sequence of nullable <b>Int64</b> values that are obtained by invoking a transform function on each element of the input sequence.
Average<TSource>(IEnumerable<TSource>, Func<TSource, Nullable<Single>>)	Computes the average of a sequence of nullable <b>Single</b> values that are obtained by invoking a transform function on each element of the input sequence.
Average<TSource>(IEnumerable<TSource>, Func<TSource, Single>)	Computes the average of a sequence of <b>Single</b> values that are obtained by invoking a transform function on each element of the input sequence.
Cast<TResult>(IEnumerable)	Casts the elements of an <b>IEnumerable</b> to the specified type.
Chunk<TSource>(IEnumerable<TSource>, Int32)	Splits the elements of a sequence into chunks of size at most <b>size</b> .
Concat<TSource>(IEnumerable<TSource>, IEnumerable<TSource>)	Concatenates two sequences.
Contains<TSource>(IEnumerable<TSource>, TSource)	Determines whether a sequence contains a specified element by using the default equality comparer.
Contains<TSource>(IEnumerable<TSource>, TSource, IEqualityComparer<TSource>)	Determines whether a sequence contains a specified element by using a specified <b>IEqualityComparer&lt;T&gt;</b> .
Count<TSource>(IEnumerable<TSource>)	Returns the number of elements in a sequence.

<code>Count&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource, Boolean&gt;)</code>	Returns a number that represents how many elements in the specified sequence satisfy a condition.
<code>DefaultIfEmpty&lt;TSource&gt;(IEnumerable&lt;TSource&gt;)</code>	Returns the elements of the specified sequence or the type parameter's default value in a singleton collection if the sequence is empty.
<code>DefaultIfEmpty&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, TSource)</code>	Returns the elements of the specified sequence or the specified value in a singleton collection if the sequence is empty.
<code>Distinct&lt;TSource&gt;(IEnumerable&lt;TSource&gt;)</code>	Returns distinct elements from a sequence by using the default equality comparer to compare values.
<code>Distinct&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, IEqualityComparer&lt;TSource&gt;)</code>	Returns distinct elements from a sequence by using a specified <code>IEqualityComparer&lt;T&gt;</code> to compare values.
<code>DistinctBy&lt;TSource, TKey&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource, TKey&gt;)</code>	Returns distinct elements from a sequence according to a specified key selector function.
<code>DistinctBy&lt;TSource, TKey&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource, TKey&gt;, IEqualityComparer&lt;TKey&gt;)</code>	Returns distinct elements from a sequence according to a specified key selector function and using a specified comparer to compare keys.
<code>ElementAt&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, Index)</code>	Returns the element at a specified index in a sequence.
<code>ElementAt&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, Int32)</code>	Returns the element at a specified index in a sequence.
<code>ElementAtOrDefault&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, Index)</code>	Returns the element at a specified index in a sequence or a default value if the index is out of range.
<code>ElementAtOrDefault&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, Int32)</code>	Returns the element at a specified index in a sequence or a default value if the index is out of range.
<code>Except&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, IEnumerable&lt;TSource&gt;)</code>	Produces the set difference of two sequences by using the default equality comparer to compare values.
<code>Except&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, IEnumerable&lt;TSource&gt;, IEqualityComparer&lt;TSource&gt;)</code>	Produces the set difference of two sequences by using the specified <code>IEqualityComparer&lt;T&gt;</code> to compare values.

<code>ExceptBy&lt;TSource,TKey&gt;</code> ( <code>IEnumerable&lt;TSource&gt;, IEnumerable&lt;TKey&gt;, Func&lt;TSource,TKey&gt;</code> )	Produces the set difference of two sequences according to a specified key selector function.
<code>ExceptBy&lt;TSource,TKey&gt;</code> ( <code>IEnumerable&lt;TSource&gt;, IEnumerable&lt;TKey&gt;, Func&lt;TSource,TKey&gt;, IEqualityComparer&lt;TKey&gt;</code> )	Produces the set difference of two sequences according to a specified key selector function.
<code>First&lt;TSource&gt;(IEnumerable&lt;TSource&gt;)</code>	Returns the first element of a sequence.
<code>First&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,Boolean&gt;)</code>	Returns the first element in a sequence that satisfies a specified condition.
<code>FirstOrDefault&lt;TSource&gt;</code> ( <code>IEnumerable&lt;TSource&gt;</code> )	Returns the first element of a sequence, or a default value if the sequence contains no elements.
<code>FirstOrDefault&lt;TSource&gt;</code> ( <code>IEnumerable&lt;TSource&gt;, TSource</code> )	Returns the first element of a sequence, or a specified default value if the sequence contains no elements.
<code>FirstOrDefault&lt;TSource&gt;</code> ( <code>IEnumerable&lt;TSource&gt;, Func&lt;TSource,Boolean&gt;</code> )	Returns the first element of the sequence that satisfies a condition or a default value if no such element is found.
<code>FirstOrDefault&lt;TSource&gt;</code> ( <code>IEnumerable&lt;TSource&gt;, Func&lt;TSource,Boolean&gt;, TSource</code> )	Returns the first element of the sequence that satisfies a condition, or a specified default value if no such element is found.
<code>GroupBy&lt;TSource,TKey&gt;</code> ( <code>IEnumerable&lt;TSource&gt;, Func&lt;TSource,TKey&gt;</code> )	Groups the elements of a sequence according to a specified key selector function.
<code>GroupBy&lt;TSource,TKey&gt;</code> ( <code>IEnumerable&lt;TSource&gt;, Func&lt;TSource,TKey&gt;, IEqualityComparer&lt;TKey&gt;</code> )	Groups the elements of a sequence according to a specified key selector function and compares the keys by using a specified comparer.
<code>GroupBy&lt;TSource,TKey,TElement&gt;</code> ( <code>IEnumerable&lt;TSource&gt;, Func&lt;TSource,TKey&gt;, Func&lt;TSource,TElement&gt;</code> )	Groups the elements of a sequence according to a specified key selector function and projects the elements for each group by using a specified function.
<code>GroupBy&lt;TSource,TKey,TElement&gt;</code> ( <code>IEnumerable&lt;TSource&gt;, Func&lt;TSource,TKey&gt;, Func&lt;TSource,TElement&gt;, IEqualityComparer&lt;TKey&gt;</code> )	Groups the elements of a sequence according to a key selector function. The keys are compared by using a comparer and each group's elements are projected by using a specified function.
<code>GroupBy&lt;TSource,TKey,TResult&gt;</code> ( <code>IEnumerable&lt;TSource&gt;, Func&lt;TSource,TKey&gt;</code> ,	Groups the elements of a sequence according to a specified key selector function and

<code>Func&lt;TKey, IEnumerable&lt;TSource&gt;, TResult&gt;</code>	Creates a result value from each group and its key.
<code>GroupBy&lt;TSource, TKey, TResult&gt;</code> <code>(IEnumerable&lt;TSource&gt;, Func&lt;TSource, TKey&gt;, Func&lt;TKey, IEnumerable&lt;TSource&gt;, TResult&gt;, IEqualityComparer&lt;TKey&gt;)</code>	Groups the elements of a sequence according to a specified key selector function and creates a result value from each group and its key. The keys are compared by using a specified comparer.
<code>GroupBy&lt;TSource, TKey, TElement, TResult&gt;</code> <code>(IEnumerable&lt;TSource&gt;, Func&lt;TSource, TKey&gt;, Func&lt;TSource, TElement&gt;, Func&lt;TKey, IEnumerable&lt;TElement&gt;, TResult&gt;)</code>	Groups the elements of a sequence according to a specified key selector function and creates a result value from each group and its key. The elements of each group are projected by using a specified function.
<code>GroupBy&lt;TSource, TKey, TElement, TResult&gt;</code> <code>(IEnumerable&lt;TSource&gt;, Func&lt;TSource, TKey&gt;, Func&lt;TSource, TElement&gt;, Func&lt;TKey, IEnumerable&lt;TElement&gt;, TResult&gt;, IEqualityComparer&lt;TKey&gt;)</code>	Groups the elements of a sequence according to a specified key selector function and creates a result value from each group and its key. Key values are compared by using a specified comparer, and the elements of each group are projected by using a specified function.
<code>GroupJoin&lt;TOuter, TInner, TKey, TResult&gt;</code> <code>(IEnumerable&lt;TOuter&gt;, IEnumerable&lt;TInner&gt;, Func&lt;TOuter, TKey&gt;, Func&lt;TInner, TKey&gt;, Func&lt;TOuter, IEnumerable&lt;TInner&gt;, TResult&gt;)</code>	Correlates the elements of two sequences based on equality of keys and groups the results. The default equality comparer is used to compare keys.
<code>GroupJoin&lt;TOuter, TInner, TKey, TResult&gt;</code> <code>(IEnumerable&lt;TOuter&gt;, IEnumerable&lt;TInner&gt;, Func&lt;TOuter, TKey&gt;, Func&lt;TInner, TKey&gt;, Func&lt;TOuter, IEnumerable&lt;TInner&gt;, TResult&gt;, IEqualityComparer&lt;TKey&gt;)</code>	Correlates the elements of two sequences based on key equality and groups the results. A specified <code>IEqualityComparer&lt;T&gt;</code> is used to compare keys.
<code>Intersect&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, IEnumerable&lt;TSource&gt;)</code>	Produces the set intersection of two sequences by using the default equality comparer to compare values.
<code>Intersect&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, IEnumerable&lt;TSource&gt;, IEqualityComparer&lt;TSource&gt;)</code>	Produces the set intersection of two sequences by using the specified <code>IEqualityComparer&lt;T&gt;</code> to compare values.
<code>IntersectBy&lt;TSource, TKey&gt;</code> <code>(IEnumerable&lt;TSource&gt;, IEnumerable&lt;TKey&gt;, Func&lt;TSource, TKey&gt;)</code>	Produces the set intersection of two sequences according to a specified key selector function.
<code>IntersectBy&lt;TSource, TKey&gt;</code> <code>(IEnumerable&lt;TSource&gt;, IEnumerable&lt;TKey&gt;, Func&lt;TSource, TKey&gt;, IEqualityComparer&lt;TKey&gt;)</code>	Produces the set intersection of two sequences according to a specified key selector function.

<code>Join&lt;TOuter,TInner,TKey,TResult&gt;(IEnumerable&lt;TOuter&gt;, IEnumerable&lt;TInner&gt;, Func&lt;TOuter,TKey&gt;, Func&lt;TInner,TKey&gt;, Func&lt;TOuter,TInner,TResult&gt;)</code>	Correlates the elements of two sequences based on matching keys. The default equality comparer is used to compare keys.
<code>Join&lt;TOuter,TInner,TKey,TResult&gt;(IEnumerable&lt;TOuter&gt;, IEnumerable&lt;TInner&gt;, Func&lt;TOuter,TKey&gt;, Func&lt;TInner,TKey&gt;, Func&lt;TOuter,TInner,TResult&gt;, IEqualityComparer&lt;TKey&gt;)</code>	Correlates the elements of two sequences based on matching keys. A specified <code>IEqualityComparer&lt;T&gt;</code> is used to compare keys.
<code>Last&lt;TSource&gt;(IEnumerable&lt;TSource&gt;)</code>	Returns the last element of a sequence.
<code>Last&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,Boolean&gt;)</code>	Returns the last element of a sequence that satisfies a specified condition.
<code>LastOrDefault&lt;TSource&gt;(IEnumerable&lt;TSource&gt;)</code>	Returns the last element of a sequence, or a default value if the sequence contains no elements.
<code>LastOrDefault&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, TSource)</code>	Returns the last element of a sequence, or a specified default value if the sequence contains no elements.
<code>LastOrDefault&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,Boolean&gt;)</code>	Returns the last element of a sequence that satisfies a condition or a default value if no such element is found.
<code>LastOrDefault&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,Boolean&gt;, TSource)</code>	Returns the last element of a sequence that satisfies a condition, or a specified default value if no such element is found.
<code>LongCount&lt;TSource&gt;(IEnumerable&lt;TSource&gt;)</code>	Returns an <code>Int64</code> that represents the total number of elements in a sequence.
<code>LongCount&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,Boolean&gt;)</code>	Returns an <code>Int64</code> that represents how many elements in a sequence satisfy a condition.
<code>Max&lt;TSource&gt;(IEnumerable&lt;TSource&gt;)</code>	Returns the maximum value in a generic sequence.
<code>Max&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, IComparer&lt;TSource&gt;)</code>	Returns the maximum value in a generic sequence.
<code>Max&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,Decimal&gt;)</code>	Invokes a transform function on each element of a sequence and returns the maximum <code>Decimal</code> value.
<code>Max&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,Double&gt;)</code>	Invokes a transform function on each element of a sequence and returns the maximum <code>Double</code> value.

<code>Max&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,Int32&gt;)</code>	Invokes a transform function on each element of a sequence and returns the maximum <code>Int32</code> value.
<code>Max&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,Int64&gt;)</code>	Invokes a transform function on each element of a sequence and returns the maximum <code>Int64</code> value.
<code>Max&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,Nullable&lt;Decimal&gt;&gt;)</code>	Invokes a transform function on each element of a sequence and returns the maximum nullable <code>Decimal</code> value.
<code>Max&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,Nullable&lt;Double&gt;&gt;)</code>	Invokes a transform function on each element of a sequence and returns the maximum nullable <code>Double</code> value.
<code>Max&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,Nullable&lt;Int32&gt;&gt;)</code>	Invokes a transform function on each element of a sequence and returns the maximum nullable <code>Int32</code> value.
<code>Max&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,Nullable&lt;Int64&gt;&gt;)</code>	Invokes a transform function on each element of a sequence and returns the maximum nullable <code>Int64</code> value.
<code>Max&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,Nullable&lt;Single&gt;&gt;)</code>	Invokes a transform function on each element of a sequence and returns the maximum nullable <code>Single</code> value.
<code>Max&lt;TSource&gt;(IQueryable&lt;TSource&gt;, Func&lt;TSource,Single&gt;)</code>	Invokes a transform function on each element of a sequence and returns the maximum <code>Single</code> value.
<code>Max&lt;TSource,TResult&gt;(IQueryable&lt;TSource&gt;, Func&lt;TSource,TResult&gt;)</code>	Invokes a transform function on each element of a generic sequence and returns the maximum resulting value.
<code>MaxBy&lt;TSource,TKey&gt;(IQueryable&lt;TSource&gt;, Func&lt;TSource,TKey&gt;)</code>	Returns the maximum value in a generic sequence according to a specified key selector function.
<code>MaxBy&lt;TSource,TKey&gt;(IQueryable&lt;TSource&gt;, Func&lt;TSource,TKey&gt;, IComparer&lt;TKey&gt;)</code>	Returns the maximum value in a generic sequence according to a specified key selector function and key comparer.
<code>Min&lt;TSource&gt;(IQueryable&lt;TSource&gt;)</code>	Returns the minimum value in a generic sequence.
<code>Min&lt;TSource&gt;(IQueryable&lt;TSource&gt;, IComparer&lt;TSource&gt;)</code>	Returns the minimum value in a generic sequence.

<code>Min&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,Decimal&gt;)</code>	Invokes a transform function on each element of a sequence and returns the minimum <a href="#">Decimal</a> value.
<code>Min&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,Double&gt;)</code>	Invokes a transform function on each element of a sequence and returns the minimum <a href="#">Double</a> value.
<code>Min&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,Int32&gt;)</code>	Invokes a transform function on each element of a sequence and returns the minimum <a href="#">Int32</a> value.
<code>Min&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,Int64&gt;)</code>	Invokes a transform function on each element of a sequence and returns the minimum <a href="#">Int64</a> value.
<code>Min&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,Nullable&lt;Decimal&gt;&gt;)</code>	Invokes a transform function on each element of a sequence and returns the minimum nullable <a href="#">Decimal</a> value.
<code>Min&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,Nullable&lt;Double&gt;&gt;)</code>	Invokes a transform function on each element of a sequence and returns the minimum nullable <a href="#">Double</a> value.
<code>Min&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,Nullable&lt;Int32&gt;&gt;)</code>	Invokes a transform function on each element of a sequence and returns the minimum nullable <a href="#">Int32</a> value.
<code>Min&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,Nullable&lt;Int64&gt;&gt;)</code>	Invokes a transform function on each element of a sequence and returns the minimum nullable <a href="#">Int64</a> value.
<code>Min&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,Nullable&lt;Single&gt;&gt;)</code>	Invokes a transform function on each element of a sequence and returns the minimum nullable <a href="#">Single</a> value.
<code>Min&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,Single&gt;)</code>	Invokes a transform function on each element of a sequence and returns the minimum <a href="#">Single</a> value.
<code>Min&lt;TSource,TResult&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,TResult&gt;)</code>	Invokes a transform function on each element of a generic sequence and returns the minimum resulting value.
<code>MinBy&lt;TSource,TKey&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,TKey&gt;)</code>	Returns the minimum value in a generic sequence according to a specified key selector function.
<code>MinBy&lt;TSource,TKey&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,TKey&gt;, IComparer&lt;TKey&gt;)</code>	Returns the minimum value in a generic sequence according to a specified key selector function and key comparer.

<a href="#">OfType&lt;TResult&gt;(IEnumerable)</a>	Filters the elements of an <a href="#">IEnumerable</a> based on a specified type.
<a href="#">Order&lt;T&gt;(IEnumerable&lt;T&gt;)</a>	Sorts the elements of a sequence in ascending order.
<a href="#">Order&lt;T&gt;(IEnumerable&lt;T&gt;, IComparer&lt;T&gt;)</a>	Sorts the elements of a sequence in ascending order.
<a href="#">OrderBy&lt;TSource,TKey&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,TKey&gt;)</a>	Sorts the elements of a sequence in ascending order according to a key.
<a href="#">OrderBy&lt;TSource,TKey&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,TKey&gt;, IComparer&lt;TKey&gt;)</a>	Sorts the elements of a sequence in ascending order by using a specified comparer.
<a href="#">OrderByDescending&lt;TSource,TKey&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,TKey&gt;)</a>	Sorts the elements of a sequence in descending order according to a key.
<a href="#">OrderByDescending&lt;TSource,TKey&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,TKey&gt;, IComparer&lt;TKey&gt;)</a>	Sorts the elements of a sequence in descending order by using a specified comparer.
<a href="#">OrderDescending&lt;T&gt;(IEnumerable&lt;T&gt;)</a>	Sorts the elements of a sequence in descending order.
<a href="#">OrderDescending&lt;T&gt;(IEnumerable&lt;T&gt;, IComparer&lt;T&gt;)</a>	Sorts the elements of a sequence in descending order.
<a href="#">Prepend&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, TSource)</a>	Adds a value to the beginning of the sequence.
<a href="#">Reverse&lt;TSource&gt;(IEnumerable&lt;TSource&gt;)</a>	Inverts the order of the elements in a sequence.
<a href="#">Select&lt;TSource,TResult&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,TResult&gt;)</a>	Projects each element of a sequence into a new form.
<a href="#">Select&lt;TSource,TResult&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,Int32,TResult&gt;)</a>	Projects each element of a sequence into a new form by incorporating the element's index.
<a href="#">SelectMany&lt;TSource,TResult&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,IEnumerable&lt;TResult&gt;&gt;)</a>	Projects each element of a sequence to an <a href="#">IEnumerable</a> <T> and flattens the resulting sequences into one sequence.
<a href="#">SelectMany&lt;TSource,TResult&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,Int32,IEnumerable&lt;TResult&gt;&gt;)</a>	Projects each element of a sequence to an <a href="#">IEnumerable</a> <T>, and flattens the resulting sequences into one sequence. The index of

	each source element is used in the projected form of that element.
SelectMany<TSource, TCollection, TResult> (IEnumerable<TSource>, Func<TSource, IEnumerable<TCollection>>, Func<TSource, TCollection, TResult>)	Projects each element of a sequence to an <a href="#">IEnumerable&lt;T&gt;</a> , flattens the resulting sequences into one sequence, and invokes a result selector function on each element therein.
SelectMany<TSource, TCollection, TResult> (IEnumerable<TSource>, Func<TSource, Int32, IEnumerable<TCollection>>, Func<TSource, TCollection, TResult>)	Projects each element of a sequence to an <a href="#">IEnumerable&lt;T&gt;</a> , flattens the resulting sequences into one sequence, and invokes a result selector function on each element therein. The index of each source element is used in the intermediate projected form of that element.
SequenceEqual<TSource> (IEnumerable<TSource>, IEnumerable<TSource>)	Determines whether two sequences are equal by comparing the elements by using the default equality comparer for their type.
SequenceEqual<TSource> (IEnumerable<TSource>, IEnumerable<TSource>, IEqualityComparer<TSource>)	Determines whether two sequences are equal by comparing their elements by using a specified <a href="#">IEqualityComparer&lt;T&gt;</a> .
Single<TSource>(IEnumerable<TSource>)	Returns the only element of a sequence, and throws an exception if there is not exactly one element in the sequence.
Single<TSource>(IEnumerable<TSource>, Func<TSource, Boolean>)	Returns the only element of a sequence that satisfies a specified condition, and throws an exception if more than one such element exists.
SingleOrDefault<TSource> (IEnumerable<TSource>)	Returns the only element of a sequence, or a default value if the sequence is empty; this method throws an exception if there is more than one element in the sequence.
SingleOrDefault<TSource> (IEnumerable<TSource>, TSource)	Returns the only element of a sequence, or a specified default value if the sequence is empty; this method throws an exception if there is more than one element in the sequence.
SingleOrDefault<TSource> (IEnumerable<TSource>, Func<TSource, Boolean>)	Returns the only element of a sequence that satisfies a specified condition or a default value if no such element exists; this method throws an exception if more than one element satisfies the condition.

<code>SingleOrDefault&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,Boolean&gt;, TSource)</code>	Returns the only element of a sequence that satisfies a specified condition, or a specified default value if no such element exists; this method throws an exception if more than one element satisfies the condition.
<code>Skip&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, Int32)</code>	Bypasses a specified number of elements in a sequence and then returns the remaining elements.
<code>SkipLast&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, Int32)</code>	Returns a new enumerable collection that contains the elements from <code>source</code> with the last <code>count</code> elements of the source collection omitted.
<code>SkipWhile&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,Boolean&gt;)</code>	Bypasses elements in a sequence as long as a specified condition is true and then returns the remaining elements.
<code>SkipWhile&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,Int32,Boolean&gt;)</code>	Bypasses elements in a sequence as long as a specified condition is true and then returns the remaining elements. The element's index is used in the logic of the predicate function.
<code>Sum&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,Decimal&gt;)</code>	Computes the sum of the sequence of <b>Decimal</b> values that are obtained by invoking a transform function on each element of the input sequence.
<code>Sum&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,Double&gt;)</code>	Computes the sum of the sequence of <b>Double</b> values that are obtained by invoking a transform function on each element of the input sequence.
<code>Sum&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,Int32&gt;)</code>	Computes the sum of the sequence of <b>Int32</b> values that are obtained by invoking a transform function on each element of the input sequence.
<code>Sum&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,Int64&gt;)</code>	Computes the sum of the sequence of <b>Int64</b> values that are obtained by invoking a transform function on each element of the input sequence.
<code>Sum&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,Nullable&lt;Decimal&gt;&gt;)</code>	Computes the sum of the sequence of nullable <b>Decimal</b> values that are obtained by invoking a transform function on each element of the input sequence.
<code>Sum&lt;TSource&gt;(IEnumerable&lt;TSource&gt;, Func&lt;TSource,Nullable&lt;Double&gt;&gt;)</code>	Computes the sum of the sequence of nullable <b>Double</b> values that are obtained by

	invoking a transform function on each element of the input sequence.
Sum<TSource>(IEnumerable<TSource>, Func<TSource, Nullable<Int32>>)	Computes the sum of the sequence of nullable <a href="#">Int32</a> values that are obtained by invoking a transform function on each element of the input sequence.
Sum<TSource>(IEnumerable<TSource>, Func<TSource, Nullable<Int64>>)	Computes the sum of the sequence of nullable <a href="#">Int64</a> values that are obtained by invoking a transform function on each element of the input sequence.
Sum<TSource>(IEnumerable<TSource>, Func<TSource, Nullable<Single>>)	Computes the sum of the sequence of nullable <a href="#">Single</a> values that are obtained by invoking a transform function on each element of the input sequence.
Sum<TSource>(IEnumerable<TSource>, Func<TSource, Single>)	Computes the sum of the sequence of <a href="#">Single</a> values that are obtained by invoking a transform function on each element of the input sequence.
Take<TSource>(IEnumerable<TSource>, Int32)	Returns a specified number of contiguous elements from the start of a sequence.
Take<TSource>(IEnumerable<TSource>, Range)	Returns a specified range of contiguous elements from a sequence.
TakeLast<TSource>(IEnumerable<TSource>, Int32)	Returns a new enumerable collection that contains the last <code>count</code> elements from <code>source</code> .
TakeWhile<TSource>(IEnumerable<TSource>, Func<TSource, Boolean>)	Returns elements from a sequence as long as a specified condition is true.
TakeWhile<TSource>(IEnumerable<TSource>, Func<TSource, Int32, Boolean>)	Returns elements from a sequence as long as a specified condition is true. The element's index is used in the logic of the predicate function.
ToArray<TSource>(IEnumerable<TSource>)	Creates an array from a <a href="#">IEnumerable&lt;T&gt;</a> .
ToDictionary<TSource, TKey> (IEnumerable<TSource>, Func<TSource, TKey>)	Creates a <a href="#">Dictionary&lt;TKey, TValue&gt;</a> from an <a href="#">IEnumerable&lt;T&gt;</a> according to a specified key selector function.
ToDictionary<TSource, TKey> (IEnumerable<TSource>, Func<TSource, TKey>, IEqualityComparer<TKey>)	Creates a <a href="#">Dictionary&lt;TKey, TValue&gt;</a> from an <a href="#">IEnumerable&lt;T&gt;</a> according to a specified key selector function and key comparer.

ToDictionary<TSource,TKey,TElement> (IEnumerable<TSource>, Func<TSource,TKey>, Func<TSource,TElement>)	Creates a <a href="#">Dictionary&lt;TKey, TValue&gt;</a> from an <a href="#">IEnumerable&lt;T&gt;</a> according to specified key selector and element selector functions.
ToDictionary<TSource,TKey,TElement> (IEnumerable<TSource>, Func<TSource,TKey>, Func<TSource,TElement>, IEqualityComparer<TKey>)	Creates a <a href="#">Dictionary&lt;TKey, TValue&gt;</a> from an <a href="#">IEnumerable&lt;T&gt;</a> according to a specified key selector function, a comparer, and an element selector function.
ToHashSet<TSource>(IEnumerable<TSource>)	Creates a <a href="#">HashSet&lt;T&gt;</a> from an <a href="#">IEnumerable&lt;T&gt;</a> .
ToHashSet<TSource>(IEnumerable<TSource>, IEqualityComparer<TSource>)	Creates a <a href="#">HashSet&lt;T&gt;</a> from an <a href="#">IEnumerable&lt;T&gt;</a> using the comparer to compare keys.
ToList<TSource>(IEnumerable<TSource>)	Creates a <a href="#">List&lt;T&gt;</a> from an <a href="#">IEnumerable&lt;T&gt;</a> .
ToLookup<TSource,TKey> (IEnumerable<TSource>, Func<TSource,TKey>)	Creates a <a href="#">Lookup&lt;TKey, TValue&gt;</a> from an <a href="#">IEnumerable&lt;T&gt;</a> according to a specified key selector function.
ToLookup<TSource,TKey> (IEnumerable<TSource>, Func<TSource,TKey>, IEqualityComparer<TKey>)	Creates a <a href="#">Lookup&lt;TKey, TValue&gt;</a> from an <a href="#">IEnumerable&lt;T&gt;</a> according to a specified key selector function and key comparer.
ToLookup<TSource,TKey,TElement> (IEnumerable<TSource>, Func<TSource,TKey>, Func<TSource,TElement>)	Creates a <a href="#">Lookup&lt;TKey, TValue&gt;</a> from an <a href="#">IEnumerable&lt;T&gt;</a> according to specified key selector and element selector functions.
ToLookup<TSource,TKey,TElement> (IEnumerable<TSource>, Func<TSource,TKey>, Func<TSource,TElement>, IEqualityComparer<TKey>)	Creates a <a href="#">Lookup&lt;TKey, TValue&gt;</a> from an <a href="#">IEnumerable&lt;T&gt;</a> according to specified key selector function, a comparer and an element selector function.
TryGetNonEnumeratedCount<TSource> (IEnumerable<TSource>, Int32)	Attempts to determine the number of elements in a sequence without forcing an enumeration.
Union<TSource>(IEnumerable<TSource>, IEnumerable<TSource>)	Produces the set union of two sequences by using the default equality comparer.
Union<TSource>(IEnumerable<TSource>, IEnumerable<TSource>, IEqualityComparer<TSource>)	Produces the set union of two sequences by using a specified <a href="#">IEqualityComparer&lt;T&gt;</a> .
UnionBy<TSource,TKey> (IEnumerable<TSource>, Func<TSource,TKey>)	Produces the set union of two sequences according to a specified key selector function.

<code>UnionBy&lt;TSource,TKey&gt;</code> <code>(IEnumerable&lt;TSource&gt;,</code> <code>IEnumerable&lt;TSource&gt;, Func&lt;TSource,TKey&gt;,</code> <code>IEqualityComparer&lt;TKey&gt;)</code>	Produces the set union of two sequences according to a specified key selector function.
<code>Where&lt;TSource&gt;(IQueryable&lt;TSource&gt;,</code> <code>Func&lt;TSource,Boolean&gt;)</code>	Filters a sequence of values based on a predicate.
<code>Where&lt;TSource&gt;(IQueryable&lt;TSource&gt;,</code> <code>Func&lt;TSource,Int32,Boolean&gt;)</code>	Filters a sequence of values based on a predicate. Each element's index is used in the logic of the predicate function.
<code>Zip&lt;TFirst,TSecond&gt;(IQueryable&lt;TFirst&gt;,</code> <code>IQueryable&lt;TSecond&gt;)</code>	Produces a sequence of tuples with elements from the two specified sequences.
<code>Zip&lt;TFirst,TSecond,TThird&gt;</code> <code>(IQueryable&lt;TFirst&gt;, IQueryable&lt;TSecond&gt;,</code> <code>IEnumerable&lt;TThird&gt;)</code>	Produces a sequence of tuples with elements from the three specified sequences.
<code>Zip&lt;TFirst,TSecond,TResult&gt;</code> <code>(IQueryable&lt;TFirst&gt;, IQueryable&lt;TSecond&gt;,</code> <code>Func&lt;TFirst,TSecond,TResult&gt;)</code>	Applies a specified function to the corresponding elements of two sequences, producing a sequence of the results.
<code>AsParallel(IQueryable)</code>	Enables parallelization of a query.
<code>AsParallel&lt;TSource&gt;(IQueryable&lt;TSource&gt;)</code>	Enables parallelization of a query.
<code>AsQueryable(IQueryable)</code>	Converts an <code>IEnumerable</code> to an <code>IQueryable</code> .
<code>AsQueryable&lt;TElement&gt;</code> <code>(IQueryable&lt;TElement&gt;)</code>	Converts a generic <code>IEnumerable&lt;T&gt;</code> to a generic <code>IQueryable&lt;T&gt;</code> .
<code>Ancestors&lt;T&gt;(IQueryable&lt;T&gt;)</code>	Returns a collection of elements that contains the ancestors of every node in the source collection.
<code>Ancestors&lt;T&gt;(IQueryable&lt;T&gt;, XName)</code>	Returns a filtered collection of elements that contains the ancestors of every node in the source collection. Only elements that have a matching <code>XName</code> are included in the collection.
<code>DescendantNodes&lt;T&gt;(IQueryable&lt;T&gt;)</code>	Returns a collection of the descendant nodes of every document and element in the source collection.
<code>Descendants&lt;T&gt;(IQueryable&lt;T&gt;)</code>	Returns a collection of elements that contains the descendant elements of every element and document in the source collection.
<code>Descendants&lt;T&gt;(IQueryable&lt;T&gt;, XName)</code>	Returns a filtered collection of elements that contains the descendant elements of every

	element and document in the source collection. Only elements that have a matching <a href="#">XName</a> are included in the collection.
<a href="#">Elements&lt;T&gt;(IEnumerable&lt;T&gt;)</a>	Returns a collection of the child elements of every element and document in the source collection.
<a href="#">Elements&lt;T&gt;(IEnumerable&lt;T&gt;, XName)</a>	Returns a filtered collection of the child elements of every element and document in the source collection. Only elements that have a matching <a href="#">XName</a> are included in the collection.
<a href="#">InDocumentOrder&lt;T&gt;(IEnumerable&lt;T&gt;)</a>	Returns a collection of nodes that contains all nodes in the source collection, sorted in document order.
<a href="#">Nodes&lt;T&gt;(IEnumerable&lt;T&gt;)</a>	Returns a collection of the child nodes of every document and element in the source collection.
<a href="#">Remove&lt;T&gt;(IEnumerable&lt;T&gt;)</a>	Removes every node in the source collection from its parent node.

## Applies to

Product	Versions
<b>.NET</b>	Core 1.0, Core 1.1, Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8
<b>.NET Framework</b>	2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
<b>.NET Standard</b>	1.0, 1.1, 1.2, 1.3, 1.4, 1.6, 2.0, 2.1
<b>UWP</b>	10.0
<b>Xamarin.iOS</b>	10.8
<b>Xamarin.Mac</b>	3.0

## Thread Safety

Public static ( Shared in Visual Basic) members of this type are thread safe. Any instance members are not guaranteed to be thread safe.

It is safe to perform multiple read operations on a [List<T>](#), but issues can occur if the collection is modified while it's being read. To ensure thread safety, lock the collection during a read or write operation. To enable a collection to be accessed by multiple threads for reading and writing, you must implement your own synchronization. For collections with built-in synchronization, see the classes in the [System.Collections.Concurrent](#) namespace. For an inherently thread-safe alternative, see the [ImmutableList<T>](#) class.

## See also

- [IList](#)
- [ImmutableList<T>](#)
- [Performing Culture-Insensitive String Operations in Collections](#)
- [Iterators \(C#\)](#)
- [Iterators \(Visual Basic\)](#)

# Studio User Guide

RELEASE:

2023.4



## TABLE OF CONTENTS

### [UI Activities Properties](#)

## UI Activities Properties

There are multiple activities that can be used to automate apps or web-apps and you can find them in the **Activities** panel, under the **UI Automation** category.

All of these activities have multiple properties in common:

- **ContinueOnError** – specifies if the automation should continue, even if the activity throws an error. This field only supports boolean values (True, False). The default value in this field is False. As a result, if this field is blank and an error is thrown, the execution of the project stops. If the value is set to True, the execution of the project continues regardless of any error.

 **NOTE:**

If the **ContinueOnError** field of an activity inside a [Try Catch](#) is set to True, no error is caught when the project is executed.

- **DelayAfter** – adds a pause after the activity, in milliseconds.
- **DelayBefore** – adds a pause before the activity, in milliseconds.

- **TimeoutMS** – specifies the amount of time (in milliseconds) to wait for a specified element to be found before an error is thrown. The default value is 30000 milliseconds (30 seconds).
- **WaitForReady** - Before performing the actions, wait for the target to become ready. The following options are available:
  - **None** - Does not wait for anything except the target UI element to exist before executing the action. For example, you can use this option if you want to retrieve just text from a web page or click a particular button, without having to wait for all UI elements to load. Note that this may have unwanted consequences if the button relies on elements which are not yet loaded, such as scripts.
  - **Interactive** - Waits for the target app to indicate readiness, although some assets may still be loading.
  - **Complete** - Waits for all of the UI elements in the target app to load before actually executing the action.

To assess if an application is in the Interactive or Complete state, the following tags are verified:

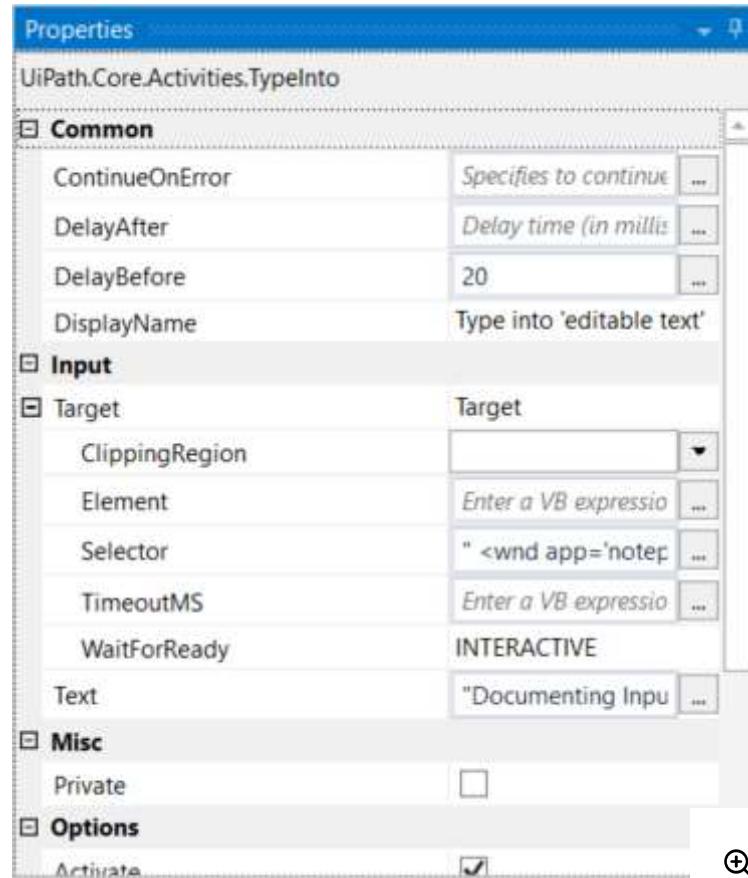
**Desktop applications** - A `wm_null` message is sent to check the existence of the `<wnd>`, `<ctrl>`, `<java>`, or `<uia>` tags. If they exist, the activity is executed.

#### Web applications:

- **Internet Explorer** - The `<webctrl>` tag is used to check if the **Ready** state of the HTML document is set to **Complete**. Additionally, the **Busy** state has to be set to "False".
- **Others** - The `<webctrl>` tag is used to check if the **Ready** state of the HTML document is **Complete**.

**SAP applications** - First the presence of the `<wnd>` tag verified, after which a SAP specific API is used to detect if the session is busy or not.

- **Target** – identifies the UI element the activity works with.



The target is composed of multiple pieces, namely the container, selector and clipping region, to ensure that you correctly identify a UI element.

A container gives you a little more context for the button or field you want to use, so that you can tell windows apart or different areas of the same app. They are automatically generated, but you can make changes to them in the **Properties** panel.

The following are containers:

- [Attach Window](#)
- [Open Application](#)
- [Attach Browser](#)
- [Open Browser](#)
- [Get Active Window](#)



Default Theme

English

# Studio User Guide

RELEASE: 2023.4



## TABLE OF CONTENTS

### [Managing Variables](#)

[Variable Type](#)

[Variable Scope](#)

[Creating Variables](#)

[From the Data Manager](#)

[From the Body of an Activity](#)

[From the Properties Panel](#)

[From the Variables Panel](#)

[Removing Variables](#)

[From the Data Manager](#)

[From the Variables Panel](#)

[Removing All Unused Variables](#)

[Browsing for .Net Variable Types](#)

# Managing Variables

In Studio, variables are used to store multiple types of data. Another key aspect of variables is that their value can change so that you can, for example, control how many times the body of a loop is executed.

 **NOTE:**

- Variables need to be created with different names, even if used in different Scopes. You can check out our [Workflow Design Naming Conventions](#) recommendations.
- Variable names must start with a letter or underscore \_.

From Studio's perspective, the variable's name is its unique ID and it defines the way it's being displayed and used. If that changes or gets deleted, Studio is not able to interpret the .xaml files based on the loaded activities. All strings have to be placed between quotation marks.

**⚠️ IMPORTANT:**

If there are both a **variable** and an **argument** with the same name, the variable is always defaulted to and used at runtime.

You can manage variables in multiple ways. To benefit from the best experience and flexibility, we recommend using the [Data Manager](#).

## Variable Type

The data stored within a variable is called a value, and it can be of multiple types. When you create a variable, the following options are available:

- **Boolean** - This type has only two possible values: `true` or `false`. These variables enable you to make decisions, and therefore have a better control over your workflow.
- **Int32** - This type is used to store numeric information. Some of its uses are to perform equations, comparisons and pass important data.
- **String** - Stores text information. This type of variable can be used to store any text-based information such as employee names, usernames or any other strings.
- **Object** - Can be used to store different types of data. You can use this type of variable in situations where you need flexibility for the type of data you store in a variable. Make sure that the data type used in Object variables is compatible with the activities that use the variable.
- **System.Data.DataTable** - This type can store big pieces of information, and act as a database or a simple spreadsheet with rows and columns. Can be useful to migrate specific data from a database to another and extract information from a website and store it locally in a spreadsheet, among other uses.
- **Array of [T]** - Enables you to store multiple values of the same type.
- **Browse for Types** - Allows you to [explore more types](#) that can be relevant to your task.

**ⓘ NOTE:**

Using variables of types defined by the language used in the project as static results in a compilation error in Windows and cross-platform projects.

## Variable Scope

The scope property gives variables a well-defined context in which they can be used. The scope can be set to global (available in the entire project), the current workflow file, or any container activity within the workflow file.

You can only create global variables from the Data Manager.

**ⓘ NOTE:**

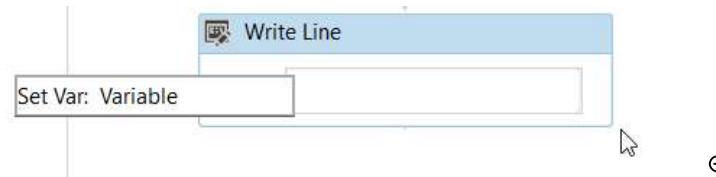
Global variables are not serializable, and therefore incompatible with long-running persistence activities.

## Creating Variables

### From the Data Manager

1. Open the Data Manager, then select **New > New Variable**.
2. Configure the name, data type, scope, and default value for the variable.

### From the Body of an Activity



1. From the **Activities** panel, drag an activity to the **Designer** panel. Either right-click a field and select **Create Variable** from the context menu, press **Ctrl+K**, or select **Create Variable** from the **Plus** menu on the right side of the field. The **Set Var** field is displayed.
2. Fill in the name and press Enter. The variable is created and visible in the field. Check its scope and type in the **Data Manager**.

## From Expressions

Alternatively, variables can be created from expressions directly in an activity input field or the **Expression Editor**:

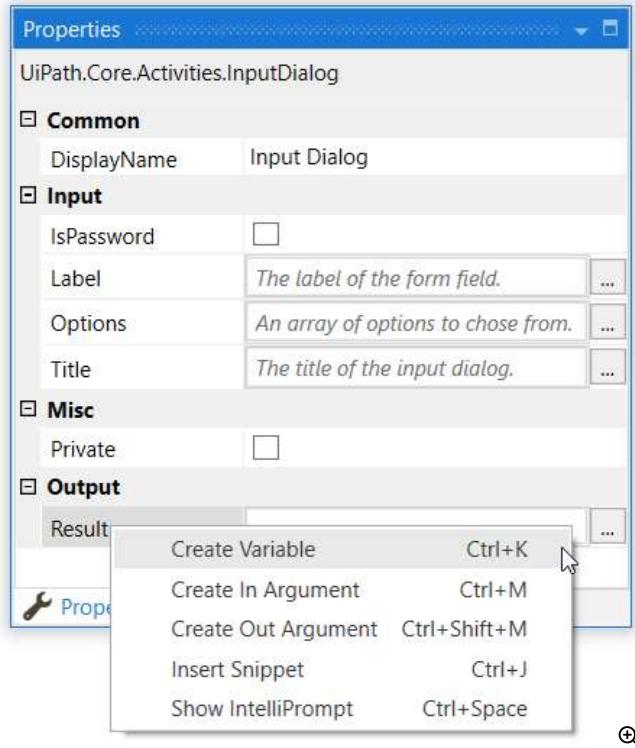
1. Select a part of the expression and press **Ctrl+K**. The **Set Var** field is displayed.
2. Fill in the name and press Enter. The variable is created. Check its scope and type in the **Data Manager**.

Name	Variable type	Scope	Default
Create Variable	Text (String)		

Variables created in these two ways automatically receive the type according to the activity. For example, if you create a variable in the **Data Table** field of a [Write Range](#) activity, the variable's type is set to **DataTable**. If you create a variable in the **Text** field of a [Write Line](#) activity, the variable's type is set to **String**.

The scope of such variables is the smallest container it is part of. The type is automatically generated depending on the selected property.

## From the Properties Panel



1. In the **Properties** panel of any activity, either right-click a field and select **Create Variable** from the context menu, press **Ctrl+K**, or select **Create Variable** from the **Plus** menu on the right side of the field. The **Set Var** field is displayed.

2. Fill in the name and press Enter. The variable is created and visible in the field. Check its scope and type in the **Data Manager**.

The scope of variables created in this way is the smallest container it is part of. The type is automatically generated depending on the selected property.

## From the Variables Panel

Name	Variable type	Scope	Default
FirstNumber	Int32	Flowchart	Enter a VB expression
SecondNumber	Int32	Flowchart	Enter a VB expression
<b>Create Variable</b>			

Variables Arguments Imports

1. In the **Designer** panel, click **Variables**. The **Variables** panel is displayed.

2. Click the **Create Variable** line, and fill in the name. A new variable is created.

The default type of variables created this way is **String**.

**Variables automatically generated from activity outputs (for cross-platform projects only)**

If the **Auto-generate Activity Outputs** design setting is enabled, Studio automatically populates each **Output** field of activities that generate an output with a variable. You can then use the generated variable in other activities by selecting the **Plus**  menu on the right side of the activity input fields and then **Use Variable**.

If you prefer to manually create an output variable, you can do so inside the output field. This will replace the generated variable with the variable that you created. If you select the **Clear value** option after manually creating a variable, the default generated variable is restored.

The generated variable is available in the **Variables** panel and in the **Data Manager** (if the variable is used in another activity). As with any other variable, you can change the generated variable's name, data type, scope, and default value.

The scope of the generated variable is based on the activity's location in the workflow (for example, the scope is different if the activity is in the main sequence or inside a **For each** activity). The scope of the generated variable updates if you copy, cut, or paste the activity. Renaming an activity does not impact the generated variable. Deleting an activity, however:

- Clears the value of the variable if the variable is not used.
- Deletes the variable if the variable is used and a warning is displayed in the activities that used the variable as an input.

## Removing Variables

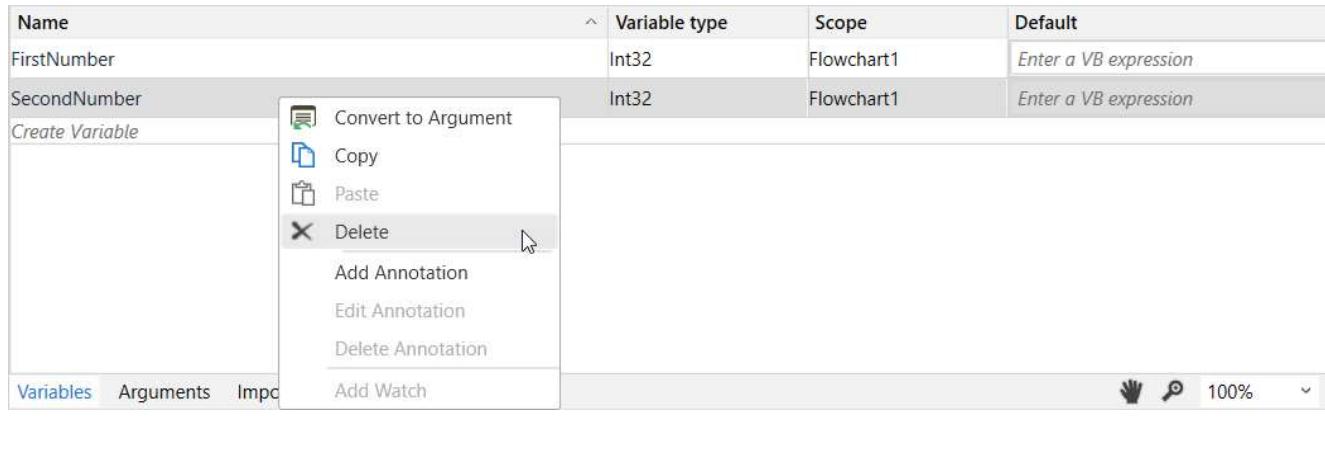
You can remove variables from the Data Manager or from the Variables panel.

### From the Data Manager

1. Open the Data Manager and expand the **\*Variables** node.
2. Right-click a variable, and then select **Delete Variable**. Alternatively, select it and press the Delete key.

### From the Variables Panel

1. Open the Variables panel.
2. Right-click a variable, and then select **Delete**. Alternatively, select it and press the Delete key.



## Removing All Unused Variables

To remove all the variables that are defined but not used anywhere in the current file, select **Remove Unused > Variables** in the Studio ribbon, and then select **Yes** when prompted for confirmation. This also removes variables that are unused but mentioned in annotations.

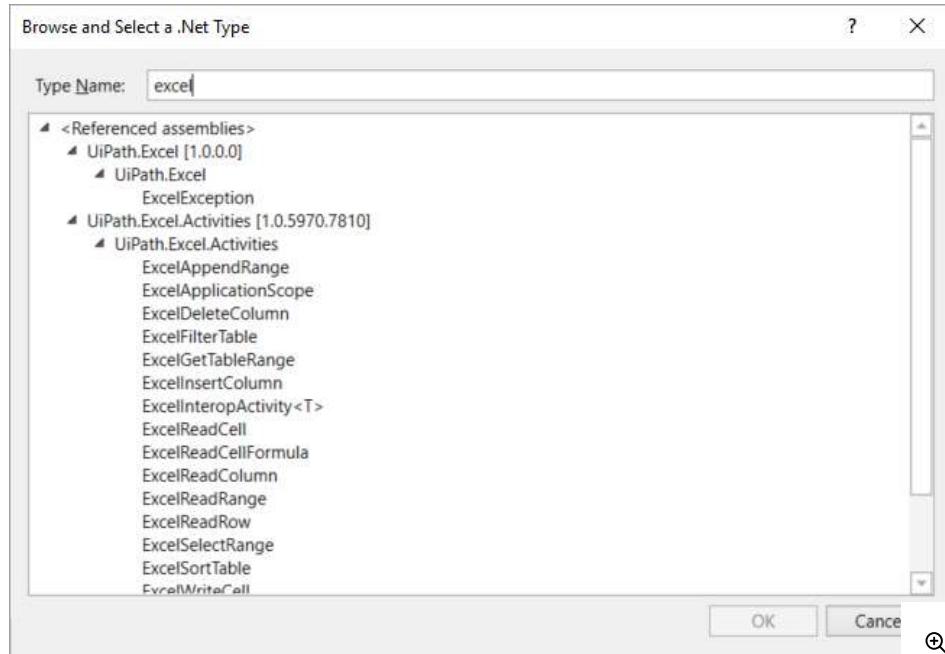
#### NOTE:

If you remove or upgrade a package that may lead to a variable or argument being undefined, an **Unknown Type** is added in its place so that the file can be opened and edited in Studio.

# Browsing for .Net Variable Types

To search for types of variables that are not displayed by default in the **Variable Type** list, do the following:

1. In the **Data Manager**, from the **Data Type** dropdown for a variable, select **Browse for Types**. You can also access the menu from the **Variable Type** dropdown in the **Variables** panel. The **Browse and Select a .Net Type** window is displayed.
2. In the **Type Name** field, type a keyword for the variable you are looking for, such as excel. Note that the result section is updated, displaying all the .Net variable types that contain your keyword.
3. Select one and click **OK**. A new variable is created with the selected type and is displayed in the **Variables** panel.



**NOTE:**

After first using a type of variable from the **Browse and Select a .Net Type** window, it is displayed in the **Variable Type** drop-down list, in the **Variables** panel.

Variable and argument types part of assemblies proprietary to Studio or Robot are hidden. Workflows that reference types from such assemblies should not be affected at runtime. To use a type from a non-.NET framework assembly, add it as a dependency to your project with the help of the [Manage Packages](#) window.

**NOTE:**

Assigning a floating point number literal (for example, 10.5) to a variable or an argument of type `float` (`System.Single`) results in a compilation error when running the workflow. To successfully assign the value, you can either:

- Use the `f` or `F` suffix (`1.5f` or `1.5F`)
- Use cast operators (`((float)1.5` for C# or `CSng(1.5)` for VB)

Default Theme

English

# Workflow Activities

## TABLE OF CONTENTS

### [Merge Data Table](#)

Project compatibility

Cross - Platform Configuration

Windows, Windows - Legacy Configuration

Example of using the Merge Data Table activity

## Merge Data Table

`UiPath.Core.Activities.MergeDataTable`

Merges the Destination with the Source, indicating whether to preserve changes and how to handle missing schema in the Source.

## Project compatibility

[Windows - Legacy](#) | [Windows](#) | [Cross-platform](#)

# Cross - Platform Configuration

- **Source Data Table** - The DataTable object to be added to the destination DataTable.
- **Destination Data Table** - The DataTable object to which the source DataTable object is merged.

## Properties

- **Missing Schema Action** - Specifies the action to take when merging the two DataTables. The options are:
  - Add
  - Ignore
  - Error
  - AddWithKey

# Windows, Windows - Legacy Configuration

## Properties

- **DisplayName** - The display name of the activity.
- **Destination** - The DataTable object to which the source DataTable is merged.
- **MissingSchemaAction** - Specifies the action to take if there are schema issues when merging the two DataTables.
- **Source** - The DataTable object to be added to the destination DataTable.
- **Private** - If selected, the values of variables and arguments are no longer logged at Verbose level.

# Example of using the Merge Data Table activity

[Here](#) you can see how the **Merge Data Table** activity is used in an example that incorporates multiple activities.



Default Theme

English

# Workflow Activities

## TABLE OF CONTENTS

### [Join Data Tables](#)

- Description
- Project compatibility
- Cross-platform configuration
- Windows - Legacy, Windows configuration
- Example of Using the Join Data Tables Activity

## Join Data Tables

`UiPath.Core.Activities.JoinDataTables`

### Description

Combines rows from two tables by using values common to each other, according to a Join rule, which is specified in the **Join Type** property.

### Project compatibility

Windows - Legacy | Windows | Cross-platform

### Cross-platform configuration

- **Data Table 1** - The first table that you want to use in the Join operation, stored in a `DataTable` variable. This field supports only `DataTable` variables.
- **Data Table 2** - The second table that you want to use in the Join operation, stored in a `DataTable` variable. This field supports only `DataTable` variables.

**NOTE:**

The order in which the two tables are supplied is very important, because it influences the structure of the resulting table, according to the option selected in the **JoinType** property field.

- **JoinType** - The type of Join operation you want to use. The following options are available:

- **Inner** - Keep all rows from **DataTable1** and **DataTable2** which meet the Join rule. Any rows that do not meet the rule are removed from the resulting table.
- **Left** - Keep all rows from **DataTable1** and only the values from **DataTable2** which meet the Join rule. Null values are inserted into the column for the rows from **DataTable1** that don't have a match in the **DataTable2** rows.
- **Full** - Keep all rows from **DataTable1** and **DataTable2**, regardless of whether the join condition is met. Null values are added into the rows from both tables that don't have a match.

**NOTE:**

If a column from **DataTable2** shares the same name with a column from **DataTable1**, then the name of the column from **DataTable2** is changed to `[ColumnName]_1` in the resulting table. If a column with the `[ColumnName]_1` name already exists, the consecutive number that is not already in use is used instead. For example, if **DataTable1** has columns named **ID**, **ID\_1** and **ID\_2**, and **DataTable2** has a column named **ID**, after the join, the column in **DataTable2** is named **ID\_3**.

- **Join Rules** - The conditions to join the tables by. Selecting the field opens a simple Filter Builder where you can add rules that compose the filter.

## Windows - Legacy, Windows configuration

### Properties panel

#### Common

- **DisplayName** - The display name of the activity.

#### Input

- **DataTable1** - The first table that you want to use in the Join operation, stored in a **DataTable** variable. This field supports only **DataTable** variables.
- **DataTable2** - The second table that you want to use in the Join operation, stored in a **DataTable** variable. This field supports only **DataTable** variables.

**NOTE:**

The order in which the two tables are supplied is very important, because it influences the structure of the resulting table, according to the option selected in the **JoinType** property field.

#### Misc

- **Private** - If selected, the values of variables and arguments are no longer logged at Verbose level.

#### Options

- **JoinType** - The type of Join operation you want to use. The following options are available:

- **Inner** - Keep all rows from **DataTable1** and **DataTable2** which meet the Join rule. Any rows that do not meet the rule are removed from the resulting table.
- **Left** - Keep all rows from **DataTable1** and only the values from **DataTable2** which meet the Join rule. Null values are inserted into the column for the rows from **DataTable1** that don't have a match in the **DataTable2** rows.

- **Full** - Keep all rows from **DataTable1** and **DataTable2**, regardless of whether the join condition is met. Null values are added into the rows from both tables that don't have a match.

**NOTE:**

If a column from **DataTable2** shares the same name with a column from **DataTable1**, then the name of the column from **DataTable2** is changed to `[ColumnName]_1` in the resulting table. If a column with the `[ColumnName]_1` name already exists, the consecutive number that is not already in use is used instead. For example, if **DataTable1** has columns named **ID**, **ID\_1** and **ID\_2**, and **DataTable2** has a column named **ID**, after the join, the column in **DataTable2** is named **ID\_3**.

## Output

- **DataTable** - The table with the joined values, stored in a **DataTable** variable. This field supports only **DataTable** Variables.

## Join Wizard

This wizard helps you configure the properties of the **Join Data Tables** activity. It can be opened by using the **Join Wizard** button in the body of the activity in the **Designer** panel.

From the upper section of the wizard, you can select both of the **DataTable** variables you wish to use in the operation, the **Join type**, and the output variable, from the following fields:

- **Input DataTable1** - The **DataTable** variable containing the first table you want to use.
- **Input DataTable2** - The **DataTable** variable containing the second table you want to use.
- **Output DataTable** - The **DataTable** variable in which you want to store the resulting table.
- **Join Type** - The type of Join operation you want to use. The following options are available:
  - **Inner** - Keep all rows from **DataTable1** and **DataTable2** which meet the Join rule. Any rows that do not meet the rule are removed from the resulting table.
  - **Left** - Keep all rows from **DataTable1** and only the values from **DataTable2** which meet the Join rule. Null values are inserted into the column for the rows from **DataTable1** that don't have a match in the **DataTable2** rows.
  - **Full** - Keep all rows from **DataTable1** and **DataTable2**, regardless of whether the join condition is met. Null values are added into the rows from both tables that don't have a match.

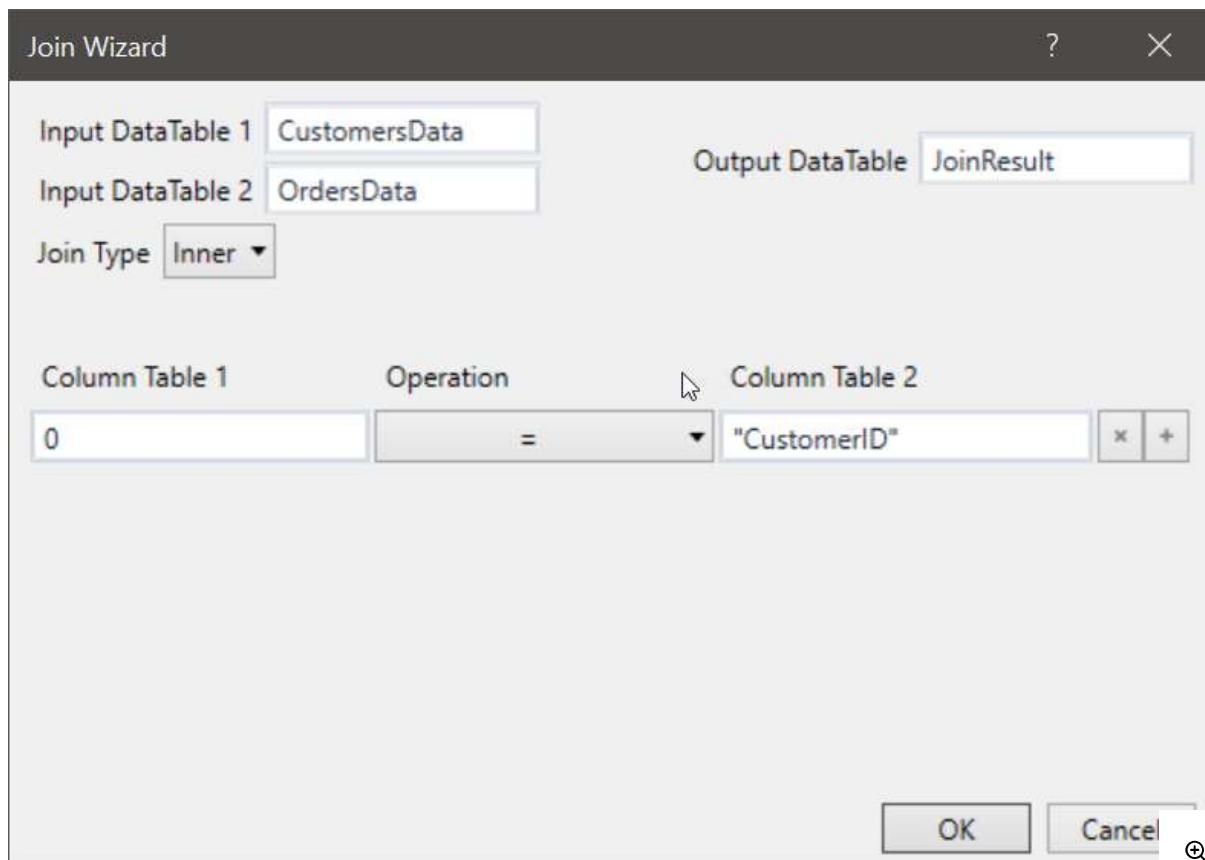
From the lower section of the wizard, you can configure the structure of the resulting table, by adding expressions that indicate relations between columns. Each of these expressions has three elements, as follows:

- **Column Table 1** - The name of the column in the first table. This field supports only **String** variables containing the column name, **Int32** variables containing the column index or **ExcelColumn** variables.
- **Operation** - The operation that defines the relation between the columns. The following options are available:
  - `=` - Equal to
  - `!=` - Not equal to
  - `>` - Greater than
  - `<` - Less than
  - `>=` - Greater than or equal to
  - `<=` - Less than or equal to
- **Column Table 2** - The name of the column in the second table. This field supports only **String** variables containing the column name, **Int32** variables containing the column index or **ExcelColumn** variables.

## Example of Using the Join Data Tables Activity

To exemplify how to use this activity, we have created a project which joins two sheets of a workbook in another sheet of that workbook. The first sheet contains details about orders, while the second sheet contains details about customers. The two sheets have a common column, **CustomerID**, which is used for the join operation. The project can be downloaded [here](#).

1. Create a blank **Project**.
2. Drag a **Sequence** container into the **Designer** panel.
3. Drag an **Excel Application Scope** and place the path of the Excel workbook in the **Workbook Path** property.
4. Create two **DataTable** variables, one for the **Customers** sheet and one for the **Orders** sheet.
5. Inside the **Excel Application Scope**, drag two **Read Range** activities.
6. Set the two **Read Range** activities to read each of the sheets in the Excel workbook and store them into their corresponding variables.
7. Create a **DataTable** variable to store the resulting table.
8. Drag a **Join Data Tables** activity in the scope container.
9. Click the **Join Wizard** button in the body of the activity. The **Join Data Tables Wizard** opens.
0. Set the variable containing the Customers sheet into the **Input DataTable 1** field.
1. Set the variable containing the Orders sheet into the **Input DataTable 2** field.
2. Set the variable created to store the resulting table into the **Output DataTable** field.
3. In the **Join Type** drop-down menu select **Inner**.
4. In the **Column Table 1** field, input the column which corresponds to the **CustomerID** column in the Customers sheet.
5. In the **Operation** drop-down menu, select **=**.
6. In the **Columns Table 2** field, input the column which corresponds to the **CustomerID** column in the Orders sheet. The Wizard should look like this:



7. Drag a **Write Range** activity to the **Designer** panel.
8. Configure the **Write Range** to write the **DataTable** variable containing the resulting table into a new sheet in the initial workbook.
9. The final workflow should look like this:







---

Default Theme

English

# Studio User Guide

RELEASE:

2023.4



## TABLE OF CONTENTS

### [State Machines](#)

Example of How to Use a State Machine

## State Machines

A state machine is a type of automation that uses a finite number of states in its execution. It can go into a state when it is triggered by an activity, and it exits that state when another activity is triggered.

Another important aspect of state machines are transitions, as they also enable you to add conditions based on which to jump from one state to another. These are represented by arrows or branches between states.

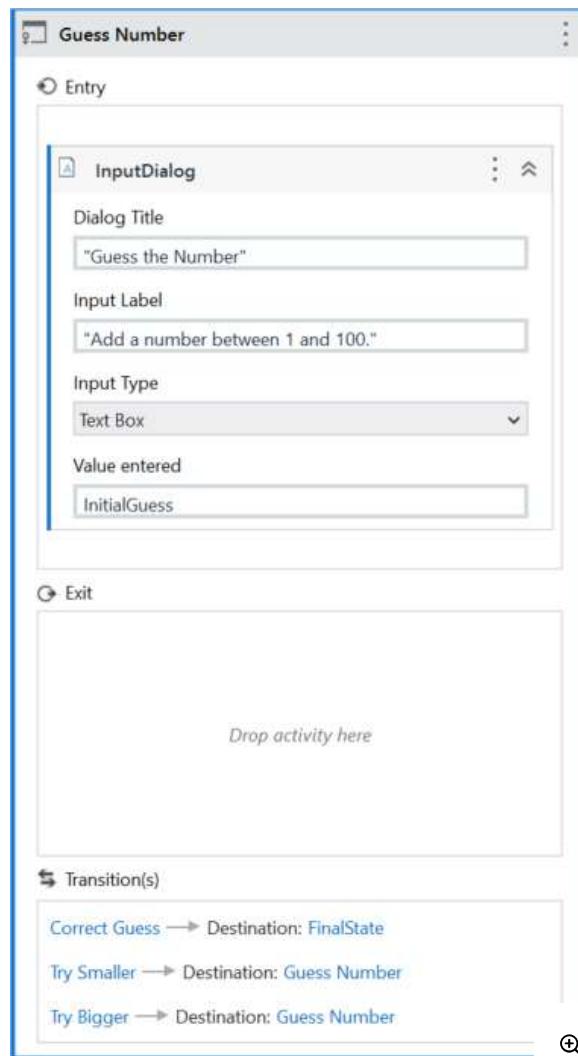
There are two activities that are specific to state machines, namely [State](#) and [Final State](#), found under **Workflow > State Machine**.

 **NOTE:**

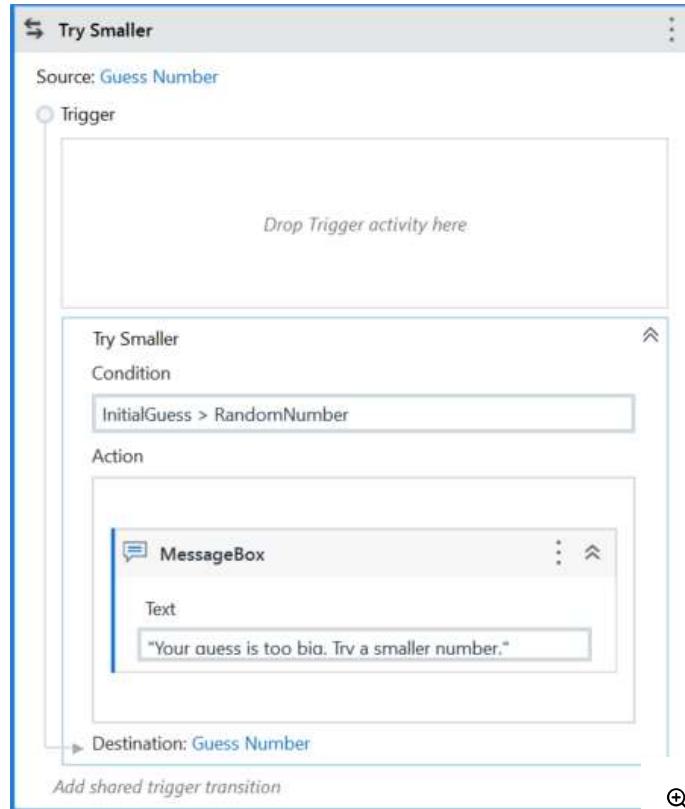
You can only create one initial state, yet it is possible to have more than one [Final State](#).

The [State](#) activity contains three sections, [Entry](#), [Exit](#) and [Transition\(s\)](#), while the [Final State](#) only contains one section, [Entry](#). Both of these activities can be expanded by double-clicking them, to view more information and edit them.

The [Entry](#) and [Exit](#) sections enable you to add entry and exit triggers for the selected state, while the [Transition\(s\)](#) section displays all the transitions linked to the selected state.



Transitions are expanded when you double-click them, just like the **State** activity. They contain three sections, **Trigger**, **Condition** and **Action**, that enable you to add a trigger for the next state, or add a condition under which an activity or sequence is to be executed.



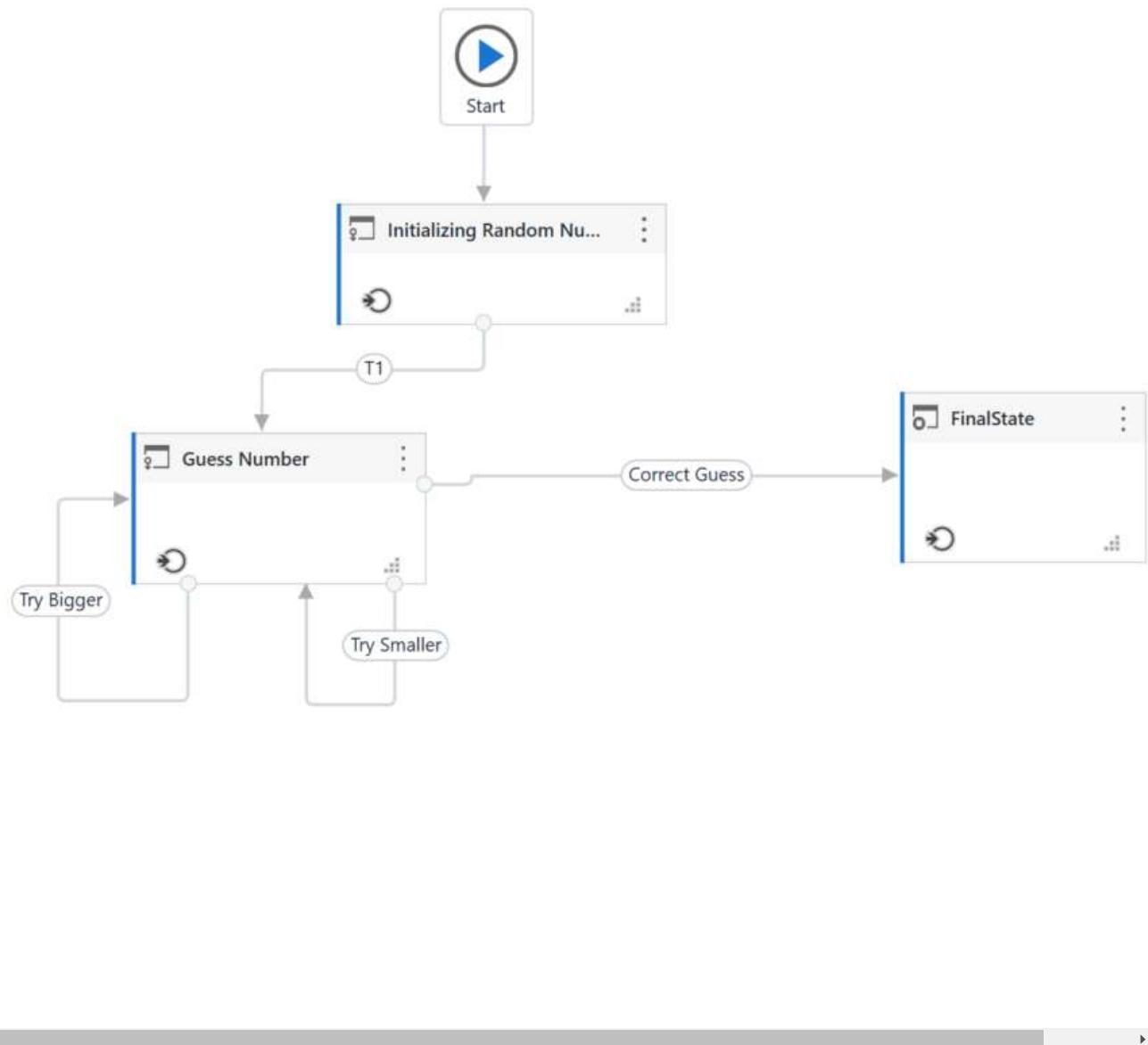
## Example of How to Use a State Machine

To exemplify how to use a state machine, we are going to build the guessing game we did in the previous chapter, the only difference being that we will try to guess a number between 1 and 100.

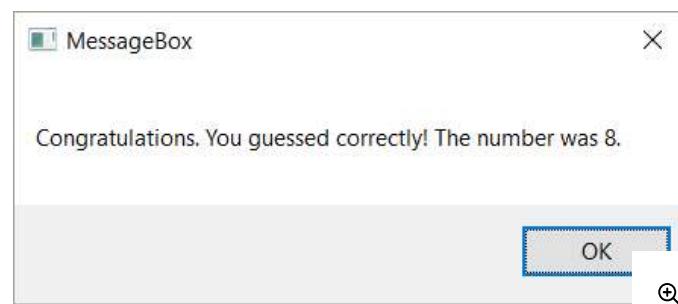
1. Create a blank process and, on the **Design** tab, in the **File** group, select **New > State Machine**. The **New State Machine** window is displayed.
- NOTE:**  
You can also add a **State Machine** activity to the **Designer** panel to create a new state machine automation.
2. In the **Name** field type a name for the automation, such as "First State Machine", and leave the default project location or add a subfolder. Click **Create**. The **Designer** panel is updated accordingly.
  3. Create two integer variables, `InitialGuess` and `RandomNumber`. The first variable stores your guess, while the second stores the random number.
  4. Add a **State** activity to the **Designer** panel and connect it to the **Start** node. This is the initial state, and it is used to generate a random number.
  5. Double-click the activity. This **State** activity is displayed expanded in the **Designer** panel.
  6. In the **Properties** panel, in the **DisplayName** field, type `Initializing Random Number`. This enables you to easily tell states apart.
  7. In the **Entry** section, add an **Assign** activity.
  8. In the **To** field, add the `RandomNumber` variable.
  9. In the **Value** field, type `new Random().Next(1,100)`. This expression generates a random number.
  0. Return to the main project view and add a new **State** activity.
  1. Connect it to the previously added activity.
  2. Double-click the last added **State** activity. This activity is displayed expanded in the **Designer** panel.
  3. In the **Properties** panel, in the **DisplayName** field, type `Guess Number`. This state is used to prompt the user to guess a number.

4. In the **Entry** section, add an [Input Dialog](#) activity.
5. Select the **Input Dialog**, and in the **Properties** panel, add an appropriate **Label** and **Title** to prompt the user to guess a number between 1 and 100.
6. In the **Result** field, add the `InitialGuess` variable. This variable stores the user's guess.
7. Return to the main project view and create a transition that points from the **Guess Number** state to itself.
8. Double-click the transition. The transition is displayed expanded in the **Designer** panel.
9. In the **Properties** panel, in the **DisplayName** field, type Try Smaller. This message is displayed on the arrow, enabling you to run through your automation easier.
20. In the **Condition** section, type `InitialGuess > RandomNumber`. This verifies if the user's guess is bigger than the random number.
21. In the **Action** section, add a [Message Box](#) activity.
22. In the **Text** field, type something similar to "Your guess is too big. Try a smaller number." This message is displayed when the user's guess is bigger than the random number.
23. Return to the main project view and create a new transition that points from the **Guess Number** state to itself.
24. Double-click the transition. The transition is displayed expanded in the **Designer** panel.
25. In the **Properties** panel, in the **DisplayName** field, type "Try Bigger". This message is displayed on the arrow, enabling you to run through your automation easier.
26. In the **Condition** section, type `InitialGuess < RandomNumber`. This verifies if the guess is smaller than the random number.
27. In the **Action** section, add a [Message Box](#) activity.
28. In the **Text** field, type something similar to "Your guess is too small. Try a bigger number." This message is displayed when the user's guess is smaller than the random number.
29. Return to main project view and add a **Final State** activity to the **Designer** panel.
30. Connect a transition from the **Guess Number** activity to the **Final State**.
31. In the **Properties** panel, in the **DisplayName** field, type "Correct Guess".
32. In the **Condition** field, type `InitialGuess = RandomNumber`. This is the condition on which this automation steps to the final state and ends.
33. Double-click the **Final State** activity. It is displayed expanded in the **Designer** panel.
34. In the **Entry** section, add a [Message Box](#) activity.
35. In the **Text** field, type something similar to "Congratulations. You guessed correctly! The number was " + `RandomNumber.ToString` + ". This is the final message that is to be displayed, when the user correctly guesses the number.

The final project should look as in the following screenshot.



36. Press F5. The automation is executed correctly.



[Download example](#)



Default Theme

English

# Studio User Guide

RELEASE: 2023.4

## TABLE OF CONTENTS

### UIExplorer

Field Descriptions for the UI Explorer Window

The Visual Tree Panel

The Selector Editor Panel

The Selector Attributes Panel

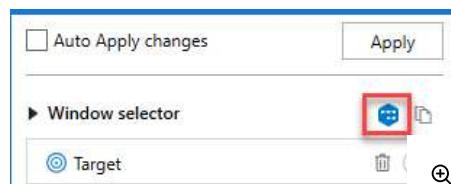
The Property Explorer Panel

## UIExplorer

**UI Explorer** is an advanced tool that enables you to create a custom selector for a specific UI element. It is available as a standalone tool you can download from the **Resource Center** in your Automation Cloud instance, or from Studio only if the `UiPath.UIAutomation.Activities` package is installed as a dependency to the project.

Using UI Explorer as a standalone tool does not require you to install Studio. This comes in handy when you want to make sure that an application can be automated with selectors. It enables you to simply inspect elements without building a process. The standalone package also contains the **SetupExtensions** utility, thus making it possible to install browser extensions and the JavaBridge to inspect elements across all your automation needs.

To open the **UI Explorer** window, click the button in the **Selectors** section, in the **Design** tab, or from the advanced editor in the selection screen of all the selectors of the **target** and **anchors** attributes, after indicating the target UI element.



#### NOTE:

If you do not have the `UiPath.UIAutomation.Activities` pack installed as a dependency for the current project, the **UI Explorer** button does not appear in the **Ribbon**.

Alternatively, the **UI Explorer** can be launched from the **Tools** page in the Studio backstage view. UI Explorer from the context menu uses the UI automation libraries shipped with the current version of Studio.

#### NOTE:

The version of the UIAutomation package that is currently used is displayed in the lower right corner in the **UI Explorer** you have opened. This version varies, as launching the **UI Explorer** from the **Tools** page uses the default **UI Automation** version shipped with the Studio version you are using, while opening **UI Explorer** from the **Ribbon** uses the version you have installed as a dependency for the current project.



To be sure that you choose the best selector, remember to:

- Add or remove attributes
- Add parent or children tags
- Use wildcards to replace changing values

**UI Explorer**

Validate Indicate Element Indicate Anchor Repair Highlight UI Frameworks

Visual Tree

Property Explorer

Selector Editor

AutomationID	Class	Name	Role
automationid	cls	num9Button	button
cls	name	Nine	
name	role		button

Target Element: 'button Nine'

UIAutomation v18.3.0

## Field Descriptions for the UI Explorer Window

Field	Description
<b>Validate</b> 	The button shows the status of the selector by checking the validity of the selector definition and the visibility of the target element on the screen. The <b>Validate</b> button has three states: <ul style="list-style-type: none"> <li>•  Validate Selector is being validated</li> <li>•  Valid selector</li> <li>•  Invalid selector</li> <li>•  Modified selector, revalidate</li> </ul>
<b>Indicate Element</b> 	Indicates a new UI element to replace the previous one.
<b>Indicate Anchor</b> 	Enables you to choose an anchor relative to the target UI element.
<b>Repair</b> 	Enables you to re-indicate the same target UI element and repair the selector. This operation does not completely replace the previous selector. The button is available only when the selector is invalid.
<b>Highlight</b> 	Brings the target element in the foreground. The highlight stays on until it's switched off. The button is enabled only if the selector is valid.
<b>UI Frameworks</b> 	Changes the technology used to determine UI elements and their selectors. The following options are available: <ul style="list-style-type: none"> <li>• Default – UiPath proprietary method. Usually works fine with all types of user interfaces.</li> <li>• Active Accessibility – an earlier solution from Microsoft for making apps accessible. It is recommended that you use this option with legacy software, when the Default one does not work.</li> <li>• UI Automation – the improved accessibility model from Microsoft. It is recommended that you use this option with newer apps, when the Default one does not work.</li> </ul> Click <a href="#">here</a> to read more about Active Accessibility and UI Automation.

## The Visual Tree Panel

Displays a tree of the UI hierarchy and enables you to navigate through it, by clicking the arrows in front of each node.

By default, the first time when you open **UI Explorer**, this panel displays all opened applications, in alphabetical order.

Double-clicking a UI element (or right-clicking and selecting **Set as Target Element**) from the tree, populates the **Selector Editor**, **Selector Attributes** and **Property Explorer** panels.

Field	Description
<b>Highlight</b> 	Highlights the selected element from the Visual Tree in real time. The highlight stays on until it's switched off.
<b>Show Search Options</b> 	Displays the search box and search filter options.
<b>Search Box</b>	<p>Enables you to look for a specific string. If an exact match is not found, nodes containing the nearest match are displayed.</p> <p>Wildcards are supported.</p> <p>Depending on the attribute selected from the Search by drop-down list, the search can be case sensitive.</p> <p><b>NOTE:</b> The search only looks for matches in the tree structure under the selected UI object.</p>
<b>Search by</b>	<p>Filters your search to a selected attribute or a selector. The contents of this drop-down list change according to the selected UI element.</p> <p><b>NOTE:</b> If Search by is set to Selector, you can only input one node in the <code>&lt;attribute name1='value1' ... /&gt;</code> format.</p>
<b>Children Only</b>	Limit your search to the first level children of the selected node. By default, this check box is not selected.

## The Selector Editor Panel

Displays the selector for the specified UI object and enables you to customize it.

The bottom part of the panel displays the actual XML fragment that you have to use in a project. Once you find the selector you want, you can copy it from here and paste it in the **Properties** panel of an activity, in the **Selector** field.

The top part of this panel enables you to view all the nodes in a selector and eliminate the ones that are not necessary by clearing the check box in front of them. An element in the list of selector nodes becomes active when you enable or disable an attribute, or when editing a selector in the bottom panel. Only one node is active at a time.

Selecting a node here displays its attributes in the **Selector Attributes** and **Property Explorer** panels.

Selectors can also be edited with the aid of variables, either by using the **CTRL+K** hotkey to create a variable in the selector itself, or by specifying an already created variable with the **CTRL+Space** combination. Using the **CTRL+K** hotkey enables you to specify a value and a name for the variable. Please note that only string variables can be used.

## The Selector Attributes Panel

Displays all the available attributes of a selected node (from the **Selector Editor** panel).

You can add or eliminate some of the node attributes by selecting or clearing the check box in front of each attribute.

Additionally, you can change the value of each attribute yet this modification is retained only if the new selector points at the originally selected UI object.

## The Property Explorer Panel

Displays all the attributes that a specified UI object can have, including the ones that do not appear in the selector. They cannot be changed.



Default Theme

English

# Workflow Activities

## TABLE OF CONTENTS

### [Retry Scope](#)

Description

Project compatibility

Cross-platform configuration

Windows - Legacy, Windows configuration

Example of Using the Retry Scope Activity

## Retry Scope

`UiPath.Core.Activities.RetryScope`

### Description

Retries the contained activities as long as the condition is not met or an error is thrown.

### Project compatibility

Windows - Legacy | Windows | Cross-platform

### Cross-platform configuration

## Advanced options

### Common

- **Continue On Error** - Specifies if the automation should continue even when the activity throws an error. This field only supports Boolean values (True, False). The default value is False. As a result, if the field is blank and an error is thrown, the execution of the project stops. If the value is set to True, the execution of the project continues regardless of any error.

 **NOTE:**

If this activity is included in **Try Catch** and the value of the **ContinueOnError** property is True, no error is caught when the project is executed.

### Options

- **NumberOfRetries** - The number of times that the sequence is to be retried.
- **RetryInterval** - Specifies the amount of time (in seconds) between each retry.

## ActivityBody

Add activities to be re-executed in this section.

## Condition

Add a condition activity in this section.

# Windows - Legacy, Windows configuration

### Properties panel

### Common

- **DisplayName** - The display name of the activity.
- **Continue On Error** - Specifies if the automation should continue even when the activity throws an error. This field only supports Boolean values (True, False). The default value is False. As a result, if the field is blank and an error is thrown, the execution of the project stops. If the value is set to True, the execution of the project continues regardless of any error.

 **NOTE:**

If this activity is included in **Try Catch** and the value of the **ContinueOnError** property is True, no error is caught when the project is executed.

### Misc

- **Private** - If selected, the values of variables and arguments are no longer logged at Verbose level.

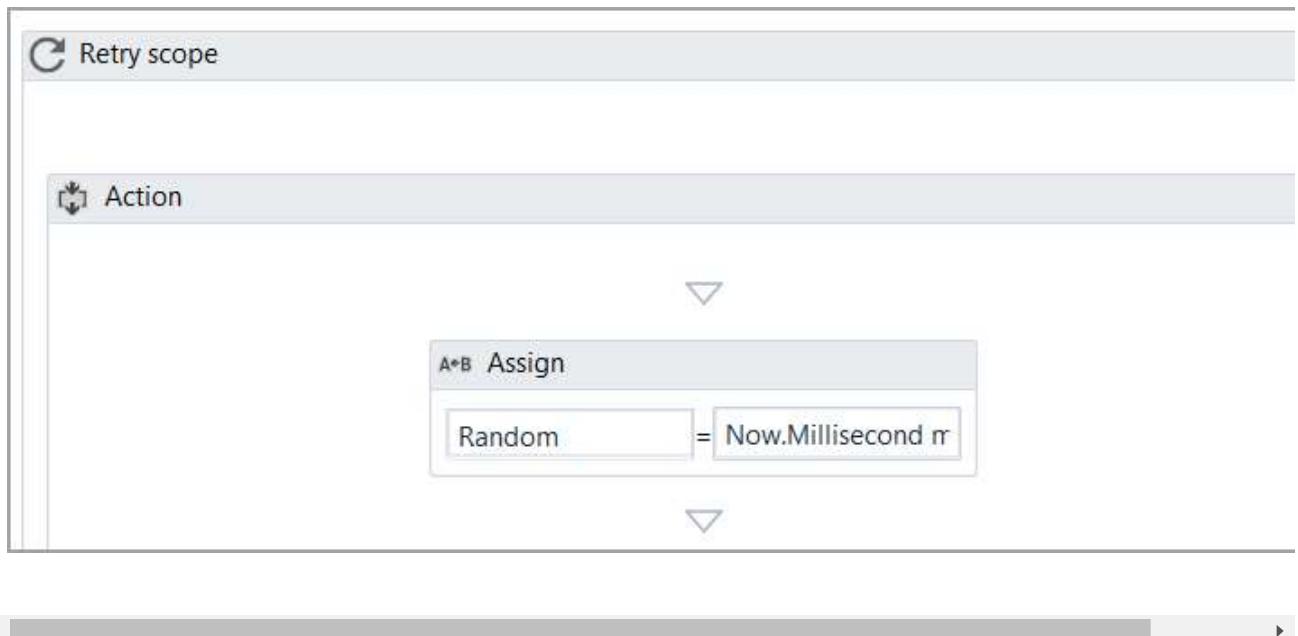
### Options

- **NumberOfRetries** - The number of times that the sequence is to be retried.
- **RetryInterval** - Specifies the amount of time (in seconds) between each retry.

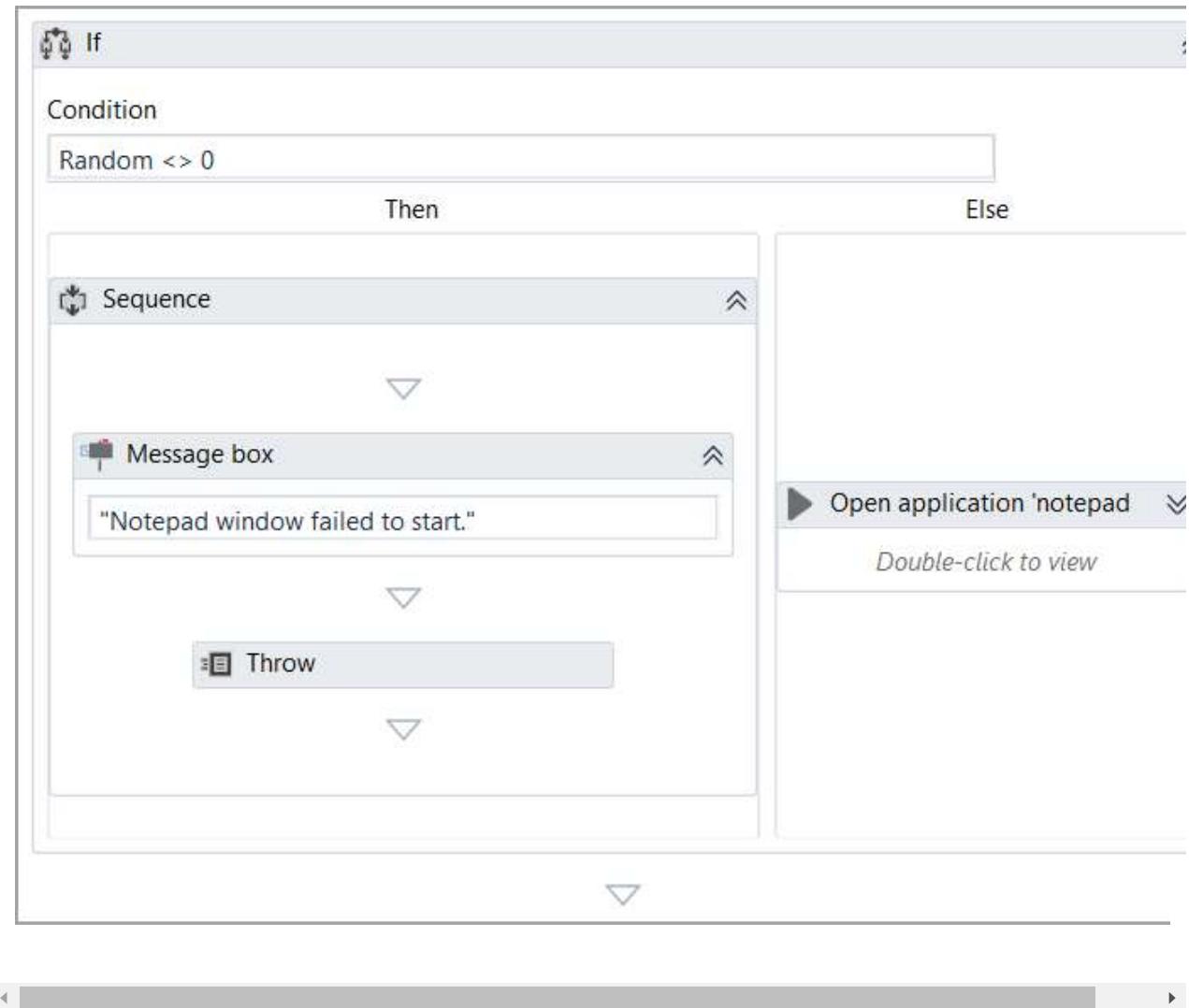
# Example of Using the Retry Scope Activity

The **Retry Scope** activity is used for catching and handling an error, which is why it's similar to the [Try Catch](#) one. The following workflow attempts to open the Notepad window 3 times and uses the condition set in the **Retry Scope** activity to stop the loop.

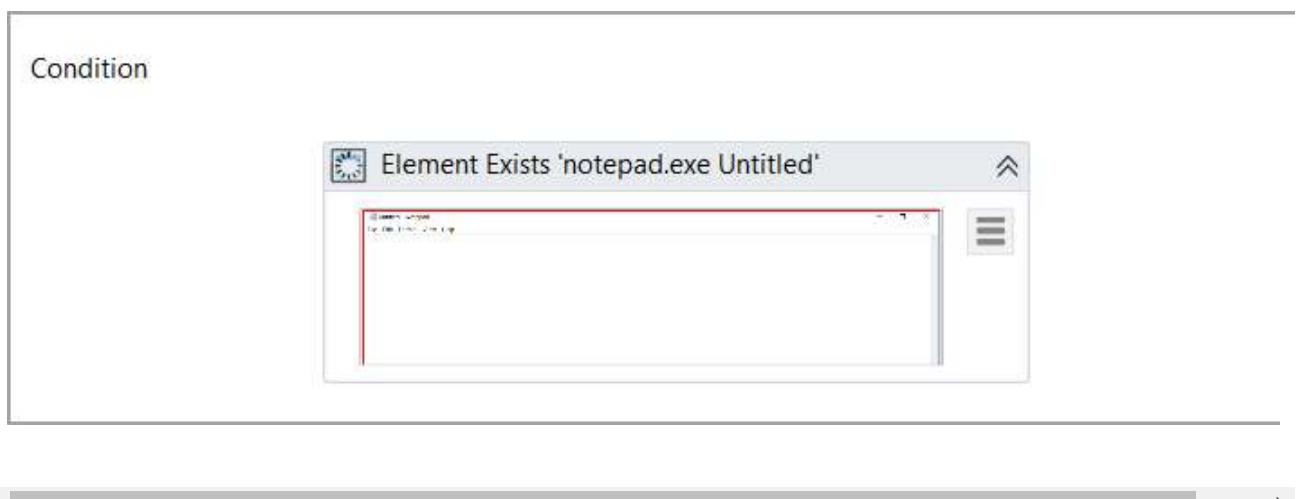
1. Create a new sequence and add the **Retry Scope** activity.
2. In the **Properties** panel, leave the default **NumberOfRetries** of 3 and the **Retry Interval** of 5. This means that we attempt to open the Notepad window 3 times and the interval between tries is 5 seconds.
3. In the **Action** section, add an **Assign** activity.
4. Create a **GenericValue** variable, named for example **Random** and add it to the **To** field of the **Assign** activity.
5. Add the `Now.Millisecond mod 5` value to the variable by adding it to the **Value** field of the **Assign** activity.



6. Add an **If** activity and as a condition enter `Random <> 0`. This means that you check if your variable is different than 0.
7. In the **Then** section of the activity (the condition above is true):
  - Add a **Message box** stating “Notepad Window failed to start”.
  - Under the **Message Box**, add a **Throw** activity to throw an error.
  - Type in `New System.Exception("Notepad failed to start")` in the **Exception** field, under **Properties**.
8. In the **Else** section of the **If** activity (the condition above is false):
  - Add an Open Application activity and indicate Notepad on the screen. Provide the full path of the Notepad executable file in the **FileName** field part of **Properties**.



9. To exit the loop, add an **Element Exists** activity in the **Condition** section of **Retry Scope** and indicate the Notepad window.



This workflow simulates a failing **Notepad** window. If the value of the Random variable is different than 0 three times in a row, the "Notepad Window failed to start" message is displayed every time and the entire workflow

fails with the “Notepad failed to start” error. The latter message is the one added in the **Throw** activity. If the value of the Random variable is 0, the Robot opens **Notepad** and because the exist condition of this loop is to find the **Notepad** window, the workflow is successfully completed.

[Download example](#)



---

Default Theme

English



# Productivity Activities

## TABLE OF CONTENTS

### [Excel Application Scope](#)

Properties

Example of Using the Excel Application Scope Activity

# Excel Application Scope

`UiPath.Excel.Activities.ExcelApplicationScope`

Opens an Excel workbook and provides a scope for Excel Activities. When the execution of this activity ends, the specified workbook and the Excel application are closed. If a `WorkbookApplication` variable is provided in the Output > Workbook property field, the spreadsheet is not closed after the activity ends. If the specified file does not exist, a new Excel file is created. This activity can only be used if the Microsoft Excel application is installed on your machine.

## Properties

 **NOTE:**

- Strings must be placed between quotation marks.

- When using Office 365, the activity opens Excel files in a new instance if Excel Application Scope is executed before manually opening the Excel files.

## Common

- DisplayName** - The display name of the activity.

## File

- Edit password** - The password required for editing password protected Excel workbooks, if necessary. Only String variables and strings are supported.
- Password** - The password required for opening password protected Excel workbooks, if necessary. Only String variables and strings are supported.
- Workbook path** - The full path of the Excel spreadsheet that you want to use. If the Excel file to be used is located in the project folder, its relative path can be used. Only String variables and strings are supported.

## Misc

- Private** - If selected, the values of variables and arguments are no longer logged at Verbose level.

## Options

- Create if not exists** - When selected, if the workbook cannot be found at the specified path, a new Excel workbook is created with the name specified in the Workbook path property field. When cleared, if the workbook cannot be found at the specified path, an exception is thrown, informing the user. By default, this checkbox is selected.
- InstanceCachePeriod** - How long to keep the Excel process alive after all child activities are executed. The default value is 3,000 ms. When the activity is used inside a loop (for example, in a **For Each** activity) or when multiple Excel Application Scope activities are used consecutively, this prevents errors that can be caused by the Excel process shutting down while opening the next file. To shut Excel down as fast as possible, set the value to 0.

### NOTE:

When working with multiple files, leaving UiPath to manage the lifetime of the Excel process creates the possibility of this **race condition** to cause an error. To prevent this, we recommend creating a parent Excel Application Scope activity, saving the output **Workbook** as a variable, placing all other Excel Application Scope activities inside this parent, and reusing the **WorkbookApplication** variable as the input for the **ExistingWorkbook** property in all child scope activities. This ensures Excel remains running for as long as the automation is using Excel.

- MacroSetting** - Specifies the macro level for the current Excel file. By default, **EnableAll** is selected. The drop-down contains three options, as follows:
  - EnableAll** - All macros are enabled and can be run.
  - DisableAll** - All macros are disabled in the specified Excel file. No macros can be run.

- **ReadFromExcelSettings** - Reads the current Excel Macro settings.
- **Read-only** - Opens the specified workbook in Read-Only mode. Selecting this check box enables you to perform data extraction operations in an Excel file which is locked for editing or has an edit password. By default, this check box is not selected.
- **Save changes** - Automatically saves the workbook on each change caused by an activity. If disabled, the changes will not be saved when the execution of the Excel Application Scope ends. By default, this checkbox is selected.
- **Visible** - When selected, the Excel file is opened in the foreground while performing actions on it. When it is cleared, all operations are done in the background.

## Output

- **Workbook** - The entire information from the Excel spreadsheet stored in a WorkbookApplication variable. This variable can be used in another Excel Application Scope activity. Only WorkbookApplication variables are supported.

## Use Existing Workbook

- **ExistingWorkbook** - Use the data from an Excel file that was previously stored in a WorkbookApplication variable. Only WorkbookApplication variables are supported.

# Example of Using the Excel Application Scope Activity

To exemplify how to use this activity, we have built an automation project which includes an **Excel Application Scope** activity among other activities. You can download the workflow [here](#).



Default Theme

English

# Productivity Activities

## TABLE OF CONTENTS

### [About the Excel Activities Package](#)

Modern Activities

Workbook Activities

App Integration Activities

Troubleshooting and Limitations

## About the Excel Activities Package

The Excel activities package aids users to automate all aspects of Microsoft Excel, as we know it is an application intensely used by many in all types of businesses.

It contains activities that enable you to read information from a cell, columns, rows or ranges, write to other spreadsheets or workbooks, execute macros, and even extract formulas. You can also sort data, color code it or append additional information.

 **NOTE:**

The **UiPath.Excel.Activities** pack is compatible with the following **Microsoft Excel** versions:

- 2013
- 2016
- 2019
- Office 365

## Modern Activities

The activities grouped under **Modern** are the newest activities that were initially designed for the StudioX profile.

The **Excel Modern** design experience and activities provide the same functionality as the StudioX business activities by default in Studio. To find out more about working with Excel activities in StudioX, see [Excel Automation](#) in the StudioX guide.

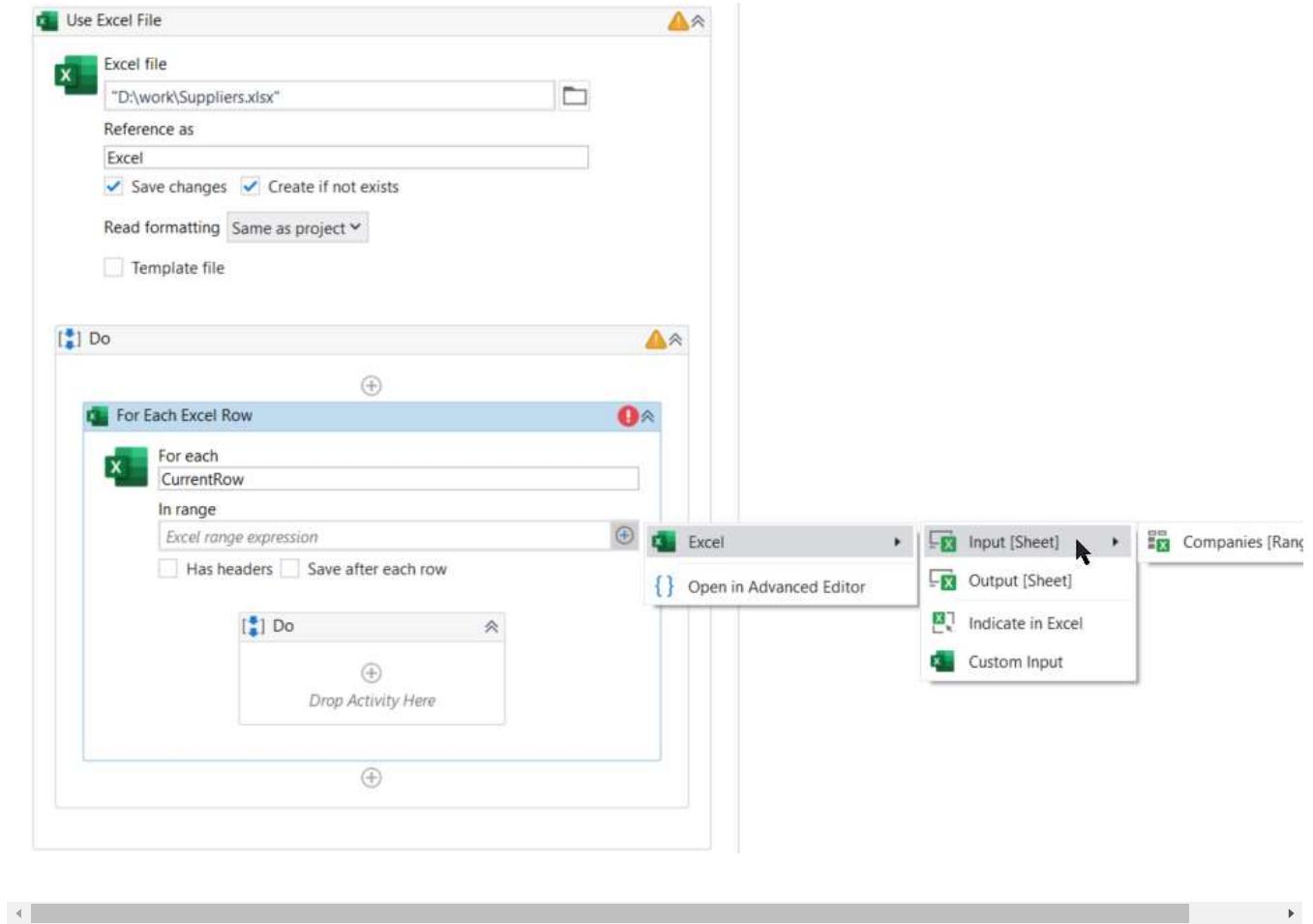
### Modern Design Experience in Studio

Starting with Studio v2021.10 and UiPath.Excel.Activities v2.11.0, a modern design experience is available in the Studio profile. In the modern experience, the modern activities replace the classic activities in the Studio profile, and the StudioX design experience is brought to Studio.

The modern design experience is enabled by default. You can select the design experience for each project from the Excel activities [project settings](#). When the classic experience is enabled for a project, the [classic app integration activities](#) that do not support interacting with Excel from the Plus  menu are available in the Activities Panel in Studio instead of the modern activities.

Add the [Use Excel File](#) scope activity to indicate the file to automate and configure its child activities by selecting Excel data from the Plus  menu of each property without having to manually enter expressions:

- Browse the contents of the file from the menu and select data that matches the type of each property. For example, you can select cells, ranges, tables, sheets, or charts from the worksheet. You can also indicate the current row or current sheet in an iteration.



- If you find it easier to work in Excel, use the **Indicate in Excel** option to select data directly from the file. This functionality requires that you install the [Excel Add-in](#).

Please select a range or table in any of the worksheets and press Confirm

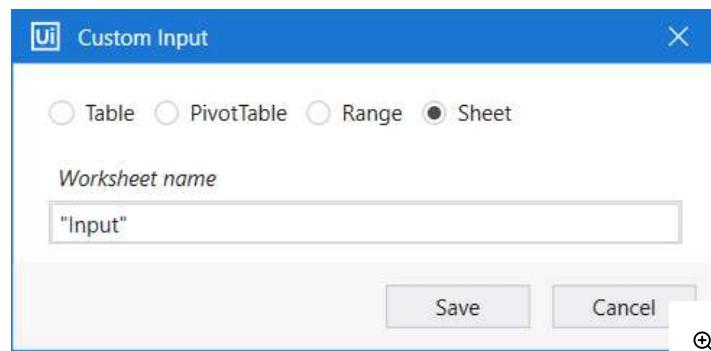
✓ Current selection: Input!A1:G10

A	B	C	D	E	F	G	H	
1	<b>Id</b>	<b>Internal Name</b>	<b>External Name</b>	<b>Website</b>	<b>Industry</b>	<b>Recurrent Supplier</b>	<b>Supplier Since</b>	<b>Number of E</b>
2	1	VPN Services	Secure VPN Provider	wwwvpn.com	Communications	TRUE	36578	
3	2	Energy Provider	ENEL	www.enel.it	Energy	TRUE	32926	
4	3	HR Provider	HR Expert	www.hrexpert.com	Human Resources	TRUE	40219	
5	4	HR Provider	HR Top Recruiters	www.hrtoprecruiters.com	Human Resources	TRUE	42776	
6	5	Internet Provider	InterPower Group	www.interpowergroup.com	Telecom	TRUE	40219	
7	6	Bank	Smart Bank	www.smartbank.com	Banking	TRUE	40413	
8	7	Wood Provider	Creative Wood	www.creativewood.com	Wood	TRUE	43506	
9	8	Metal Provider	Modern Metal	www.modernmetal.com	Metal	TRUE	43506	
10	9	Food Provider	Hungry	www.hungryfood.com	Food	TRUE	43141	
11	10	Vegetables Provider	Angry Vegetables	www.angryvegetables.com	Agriculture	TRUE	43138	
12	11	Trainers	Expert Trainers	www.experttrainers.com	Education	TRUE	43138	
13	12	Contractors 1	Teams	www.temps.com	Services	TRUE	43138	
14	13	Trainers 2	Technical Trainers	www.technicalltrainers.com	Education	TRUE	43138	
15	14	Trainers 3	Teoretic Trainers	www.teoretictrainers.com	Education	TRUE	43138	
16	15	Courses	Pragmatic	www.pragmatic.com	Education	TRUE	43138	
17	16	Contractors 3	Zodiac	www.zodiac.com	Services	TRUE	42558	

Input   Output   +

Ready   Average: 20184.94444   Count: 70   Sum: 363329

- Use the **Custom Input** option to manually specify the input based on table, chart, or sheet names or to enter references to a cell or a range.



If the file you want to automate does not exist yet, you can still use the options in the Plus menu by defining a file with the same structure as a template in the Use Excel File activity.

## Workbook Activities

[Workbook activities](#) can be executed even if Microsoft Excel is not installed on the machine and can only read/write data to and from the file.

# App Integration Activities

Classic app integration activities require the Excel app to be installed on the machine on which they run. For large and complex spreadsheets that span over a great number of rows and columns, we recommend using the App Integration activities, as these activities offer the best performance and consistency.

All app integration activities with the exception of CSV activities must be included in [Excel Application Scope](#) to work. Starting with version **2.10.4** of the Excel activities pack, these activities can also be included in the [Use Excel File](#) activity.

Dedicated activities enable you to work with .csv files, while all the others work with .xlsx and .xls. The ones that only work with [Excel Application Scope](#) can also work with .xlsm files.

## Troubleshooting and Limitations

### NOTE:

The old [Excel Binary File Format \(.xls\)](#) specific to Microsoft Excel 95 is not supported by the Excel Activities Pack.

- When you use activities that read, copy, or append ranges, please note that if a cell contains formatting information, Excel considers that cell as containing data, even though it appears empty.
- Please note that the Microsoft Office 2016 version 1708 (Build 8431.2079) changes the window title of files that are being edited and saved from "FileName.xlsx - Excel" to "FileName.xlsx - Saved". After closing the file, the filename reverts to normal.

This may cause some problems if your automation project uses selectors which contain the window title. For example, the following selector no longer works if your Excel version is the previously mentioned one: <wnd app='excel.exe' cls='XLMAIN' title='FileName.xlsx - Excel' /> <wnd cls='EXCEL7' title='FileName.xlsx' />. To prevent this issue, we recommend using a dynamic variable selector.

- Some Excel versions always open SharePoint documents in read-only mode (e.g. Excel 2019). You can check if a remote document supports editing by opening it manually in Excel on your local machine.



Default Theme

English



# Studio User Guide

RELEASE:

2023.4



## TABLE OF CONTENTS

[About Imported Namespaces](#)

## About Imported Namespaces

In UiPath Studio, namespaces represent containers that store different types of data. They enable you to easily define the scope of your expressions, variables and arguments.

For example, if you have the System.Data namespace imported, you can further use DataTable, DataView, DataColumn, DataRow and other classes that are available in it, without having to always type System.Data.DataTable and so on.

You can view and manage namespaces from the Data Manager or from the **Imports** panel. Note that some namespaces are automatically imported when you browse for a .Net type variable or argument.



Default Theme

English

# Workflow Activities

## TABLE OF CONTENTS

### [Is Match](#)

- [Description](#)
- [Project compatibility](#)
- [Cross-platform configuration](#)
- [Windows - Legacy, Windows configuration](#)
- [Example of Using the Is Match activity](#)

## Is Match

`UiPath.Core.Activities.IsMatch`

### Description

Indicates whether the specified regular expression finds a match in the specified input string, using the specified matching options. This activity has a [RegEx Builder](#) wizard that can be used to configure it, on which you can read more [here](#).

### Project compatibility

[Windows - Legacy](#) | [Windows](#) | [Cross-platform](#)

### Cross-platform configuration

- **Input:** - The string to be searched for matches.
- **Pattern** - The regular expression pattern to match.

#### Advanced options

#### Others

- **Regex Option** - A bitwise combination of the enumeration values that specify options for matching. The available options are `IgnoreCase`, `Multiline`, `ExplicitCapture`, `Compiled`, `Singeline`, `IgnorePatternWhitespace`, `RightToLeft`, `ECMAScript`, and `CultureInvariant`.
- **Result** - A boolean variable that is set to true if the regular expression finds a match, and is set to false otherwise.

## Windows - Legacy, Windows configuration

### Designer panel

- **Configure Regular Expression...** - Opens the **RegEx Builder** wizard where you can specify the regular expression pattern to be matched.

### Properties panel

#### Common

- **DisplayName** - The display name of the activity.

#### Input

- **Input:** - The string to be searched for matches.
- **Pattern** - The regular expression pattern to match.
- **RegexOption** - A bitwise combination of the enumeration values that specify options for matching.

#### Misc

- **Private** - If selected, the values of variables and arguments are no longer logged at Verbose level.
- **Result** - A boolean variable that is set to true if the regular expression finds a match, and is set to false otherwise.

## Example of Using the Is Match activity

This workflow explains how to verify the validity of an email address by using the **Is Match** activity with a custom **Regular Expression**.

This is how the automation process can be built:

1. Open Studio and create a new **Process**.
2. Drag a **Sequence** container in the Workflow Designer.

- Create the following variables:

Variable Name	Variable Type	Default Value
EmailToCheck	String	John.Doe@server.org
.IsMatch	Boolean	

3. Drag an **Is Match** activity inside the **Sequence** container.

- In the **Properties** panel, add the variable `EmailToCheck` in the **Input:** field.
- Add the variable `.IsMatch` in the **Result** field.
- Click the **Configure Regular Expression** button and customize your RegEx. For this example add the expression `^([\w\.-]+@[\\w\.-]+\.\w{2,4})$` in the **Value** field.

**NOTE:**

More information about how to customize and configure a Regular Expression can be found [here](#).

4. Drag an **If** activity below the **Is Match** activity.

- Add the variable `.IsMatch` in the **Condition** field.

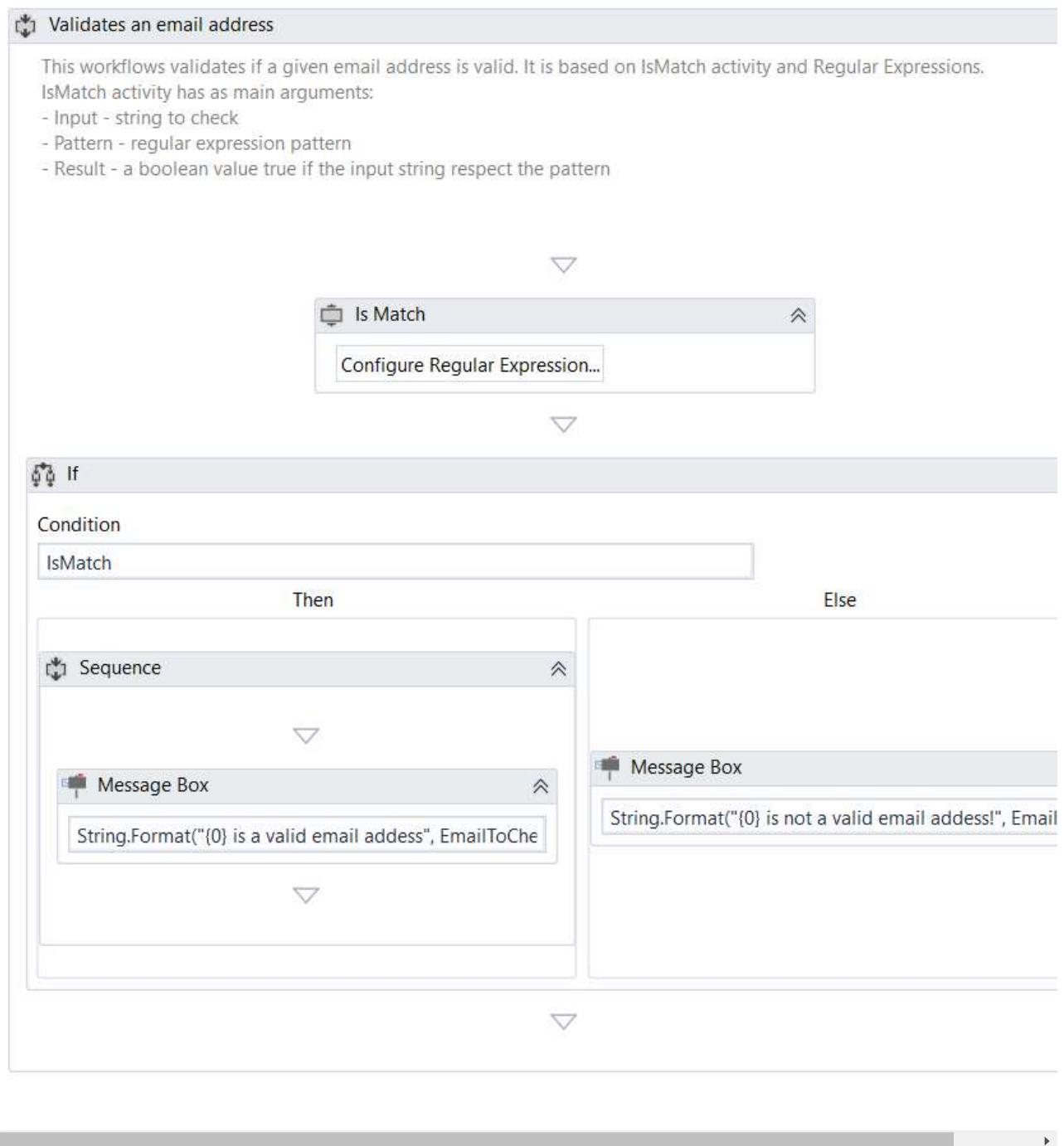
5. Drag a **Message Box** activity inside the **Then** field of the **If** activity.

- Add the expression `String.Format("{0} is a valid email address", EmailToCheck)` in the **Text** field.

6. Drag a **Message Box** activity inside the **Else** field of the **If** activity.

- Add the expression `String.Format("{0} is not a valid email address!", EmailToCheck)` in the **Text** field.

- This is how your workflow should look:



7. Run the process. The email address added as input is verified and the result is displayed in a message box.

[Download example](#)



Default Theme

English



# Workflow Activities

## TABLE OF CONTENTS

### [Matches](#)

Description

Project compatibility

Cross-platform configuration

Windows - Legacy, Windows configuration

Example of using the Matches activity

## Matches

`UiPath.Core.Activities.Matches`

## Description

Searches an input string for all occurrences of a regular expression and returns all the successful matches. This activity has a **RegEx Builder** wizard that can be used to configure it, on which you can read more [here](#).

**NOTE:**

This activity does not work well in workflows using persistence.

# Project compatibility

Windows - Legacy | Windows | Cross-platform

## Cross-platform configuration

- **Text to search in:** - The string to be searched for matches.
- **Pattern** - The regular expression pattern to match.

### Advanced options

### Others

- **Pattern Options** - A bitwise combination of the enumeration values that specify options for matching. The available options are `IgnoreCase`, `Multiline`, `ExplicitCapture`, `Compiled`, `Singeline`, `IgnorePatternWhitespace`, `RightToLeft`, `ECMAScript`, and `CultureInvariant`.

### Output

- **Result** - Reference to the collection of found [matches](#).
- **First Match** - Outputs the first match as String.

# Windows - Legacy, Windows configuration

### Designer panel

- **Configure Regular Expression...** - Opens the **RegEx Builder** wizard where you can specify the regular expression pattern to be matched.

### Properties panel

#### Common

- **DisplayName** - The display name of the activity.

### Input

- **Pattern** - The regular expression pattern to match.
- **Text to search in:** - The string to be searched for matches.

- **Pattern options** - A bitwise combination of the enumeration values that specify options for matching.

## Misc

- **Private** - If selected, the values of variables and arguments are no longer logged at Verbose level.
- **Result** - Reference to the collection of found [matches](#).

## Output

- **First Match** - Outputs the first match as String.

# Example of using the Matches activity

[Here](#) you can see how the **Matches** activity is used in an example that incorporates multiple activities.



Default Theme

English



# Workflow Activities

## TABLE OF CONTENTS

### [Match and Replace](#)

## Match and Replace

The example below explains how to find and match the values corresponding to a defined regular expression and replaces them with a specified value. This example uses activities such as the [Matches](#) and [Replace](#). You can find them in the **UiPath.System.Activities** package.

This is how the automation process can be built:

1. Open Studio and create a new **Process**.
2. Drag a **Sequence** container in the **Workflow Designer**.
  - Create the following variables:

Variable Name	Variable Type	Default Value
StringToAnalyze	String	"I have 10.5 RON in first account and 25

Variable Name	Variable Type	Default Value
		<b>RON in second account!"</b>
AccountMatches	<b>IEnumerable&lt;Match&gt;</b>	
FinalString	<b>String</b>	

3. Drag a **Matches** activity inside the **Sequence** container.

- In the **Properties** panel, add the variable **StringToAnalyze** in the **Input** field.
- Select the **IgnoreCase** and **Compiled** options from the **RegexOption** drop-down list.
- Add the variable **AccountMatches** in the **Result** field.

4. Click the **Configure Regular Expression** button.

- Select the **Advanced** option from the **RegEx** drop-down list.
- Add the expression **([0-9]+\.?[0-9]\* RON)** in the **Value** field.
- Select the **Exactly** option from the **Quantifiers** drop-down list.
- Select the check box for the **IgnoreCase** option.
- Click the **Save** button.

5. Drag a **ForEach** activity underneath the **Matches** activity.

- Inside the **ForEach** activity change the **item** to **account**.
- In the **Properties** panel, select the **System.Text.RegularExpressions.Match** option from the **TypeArgument** drop-down list.
  - Add the variable **AccountMatches** in the **Values** field.
  - Place a **Log Message** activity inside the **ForEach** activity.
  - Select the option **Info** from the **Level** drop-down list.
  - Add the expression **account.ToString** in the **Message** field.

6. Drag a **Replace** activity below the **ForEach** activity.

- In the **Properties** panel, add the variable **StringToAnalyze** in the **Input** field.
- Add the value **IgnoreCase, Compiled** in the **RegexOption** field.
- Add the expression **"\$2 \$1"** in the **Replacement** field.

- Add the variable **FinalString** in the **Result** field.

7. Click the **Configure Regular Expression** button.

- Select the **Advanced** option from the **RegEx** drop-down list.
- Add the expression `([0-9]+.*[0-9]* RON)` in the **Value** field.
- Select the **Exactly** option from the **Quantifiers** drop-down list.
- Select the check box for the **IgnoreCase** option.
- Click the **Save** button.

8. Place a **Log Message** activity inside the **ForEach** activity.

- Select the option **Info** from the **Level** drop-down list.
- Add the variable **FinalString** in the **Message** field.

9. Run the process. The robot finds and matches the values corresponding to the defined regular expression and replaces them with the specified value.

- This is how your workflow should look:



**Matches and Replace Activity**

This workflow shows the usage of Matches and Replace activities as following:

1. From an initial string it extracts all the amounts (eg. 10 RON, 12.5 RON, etc.)
2. Displays all the matches using a For Each activity
3. In the initial string, replace all the amounts with their correct formating (10 RON becomes RON 10)

▼

**Matches**

Find all the groups DDD.DDD RON where D is any digit  
The corresponding regular expression is "([0-9]+\\.[0-9]\* RON)"  
The results are outputed in a IEnumerable of Match

**Configure Regular Expression...**

▼

**Display in output pane all matches**

ForEach account in AccountMatches

**Body**

**Body**

▼

**Log Message**

Level Info

Message account.ToString

▼

**Replace**

Replace all DDDD.DDD RON with RON  
DDDD.DDD where D is any digit  
Note the usage of numbered group substitutions

Configure Regular Expression...



Log Message

Level	Info
Message	FinalString



[Here](#) you can download an example.



Default Theme

English

# Studio User Guide

RELEASE:

2023.4



## TABLE OF CONTENTS

### [Dynamic Selectors](#)

[Example of Using a Dynamic Selector](#)

# Dynamic Selectors

A dynamic selector uses a variable or an argument as a property for the attribute of your target tag. This allows the selector to easily identify a target element based on the value of the variable or argument, and not an exact string, which might change, depending on interactions inside your automation project. As such, the variable or argument can be changed to interact with a different element, without changing the selector itself. A dynamic selector has the following form, with the specifications:

- **tag** - the target tag, such as `<ctrl/>`
- **attribute** - the target attribute, such as `name='menuItem'`
- **`{{Value}}`** - the name of the variable or argument which holds the property of the element you want to interact with

#### Dynamic Selector Format:

```
<tag attribute=<code>{{Value}}</code> />
```

# Example of Using a Dynamic Selector

To illustrate the functionality of a dynamic selector, we create a simple automation process which performs a click on the **File** menu in Notepad. We then change the variable so that the **Format** menu is then clicked, but without modifying the selector.

 **NOTE:**

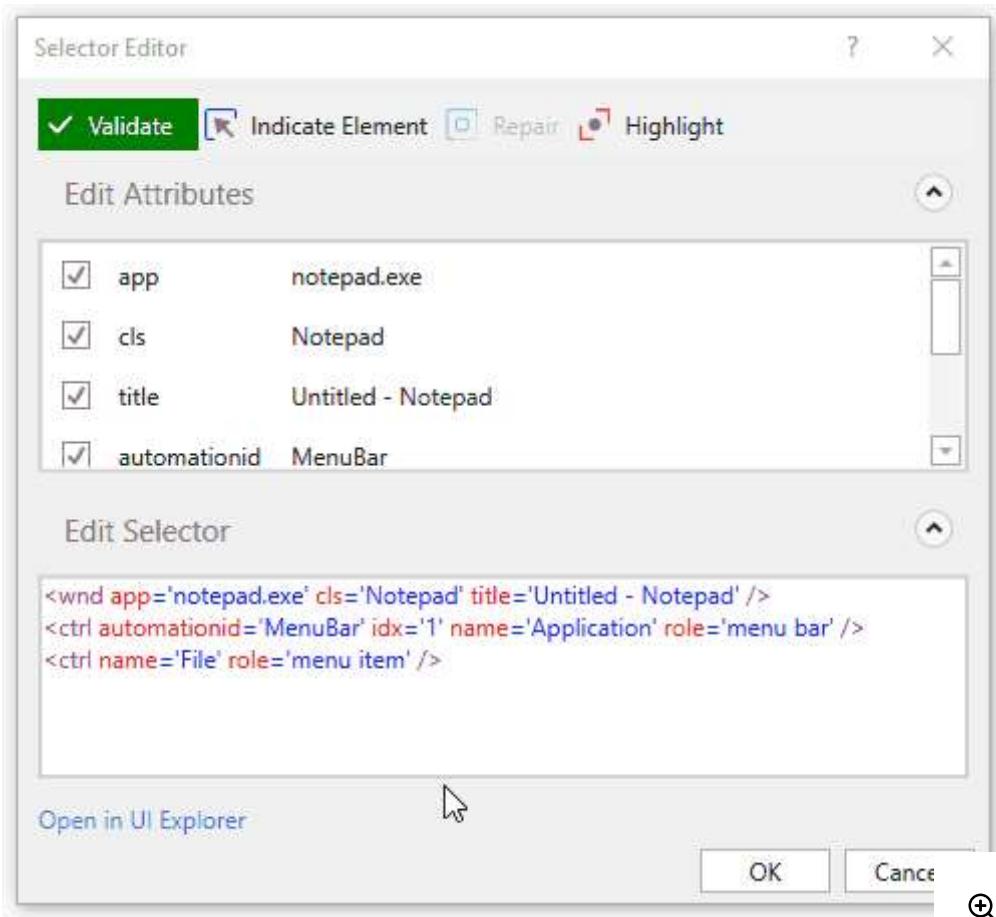
The following example uses a **variable**. Please note that **arguments** are also supported.

1. Create a new process in Studio, and add a **Click** activity.
2. Choose to **Indicate on screen** and select the **File** menu in Notepad. The following selector is generated:

```
<wnd app='notepad.exe' cls='Notepad' title='Untitled - Notepad' />
<ctrl automationid='MenuBar' idx='1' name='Application' role='menu bar' />
<ctrl name='File' role='menu item' />
```

3. Select and right-click the **File** property of the **name** attribute. A context menu with several options is displayed.
4. From the context menu, choose to **Create variable**. A couple of fields appear, allowing you to specify the variable name and value.
5. Specify a name for the new variable in the **Set Name:** field, which, in our case is **MenuOption**. Leave the **Set Value** field as is, so that the selector knows to click the attribute with the **File** property.
6. Click the **Validate** button in the **Selector Editor** window. Notice it turns green, which means our selector is valid. As such, the following dynamic selector is generated:

```
<wnd app='notepad.exe' cls='Notepad' title='Untitled - Notepad' />
<ctrl automationid='MenuBar' idx='1' name='Application' role='menu bar' />
<ctrl name='{{MenuOption}}' role='menu item' />
```



The generated selector now instructs the **Click** activity to perform the action on the **File** menu. To have it click the **Format** menu, for instance, you only need to change the default value of the variable. In this example, the operation requires a single step:

- Access the **Variables** panel in Studio and change the **Default** value of the **MenuOption** entry to **Format**. The **Click** activity now performs the action on the **Format** menu in Notepad. Note that the selector is already validated and you can start your automation process, which now clicks the **Format** menu instead of **File**.



Default Theme

English

# Studio User Guide

RELEASE:

2023.4



## TABLE OF CONTENTS

### [Selectors With Wildcards](#)

Example of Generating a Selector With Wildcards  
in the Selector Editor Window

## Selectors With Wildcards

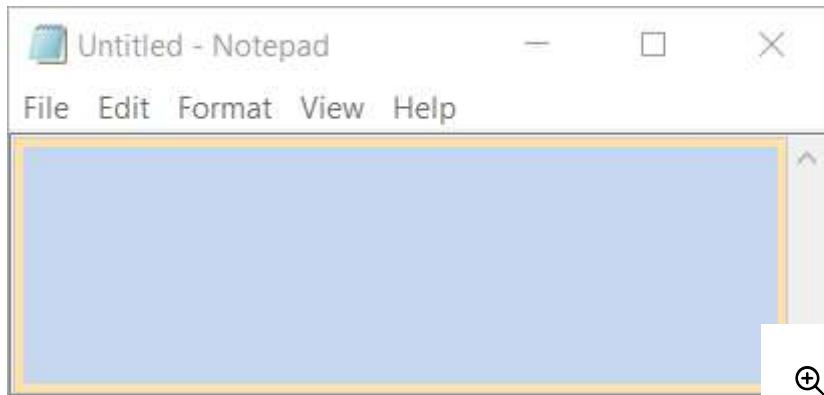
Wildcards are symbols that enable you to replace zero or multiple characters in a string. These can be quite useful when dealing with dynamically-changing attributes in a selector.

- Asterisk (\*) – replaces zero or more characters
- Question mark (?) – replaces a single character
- Asterisk AND Question mark (\*?) - matches an attribute value with at least one character

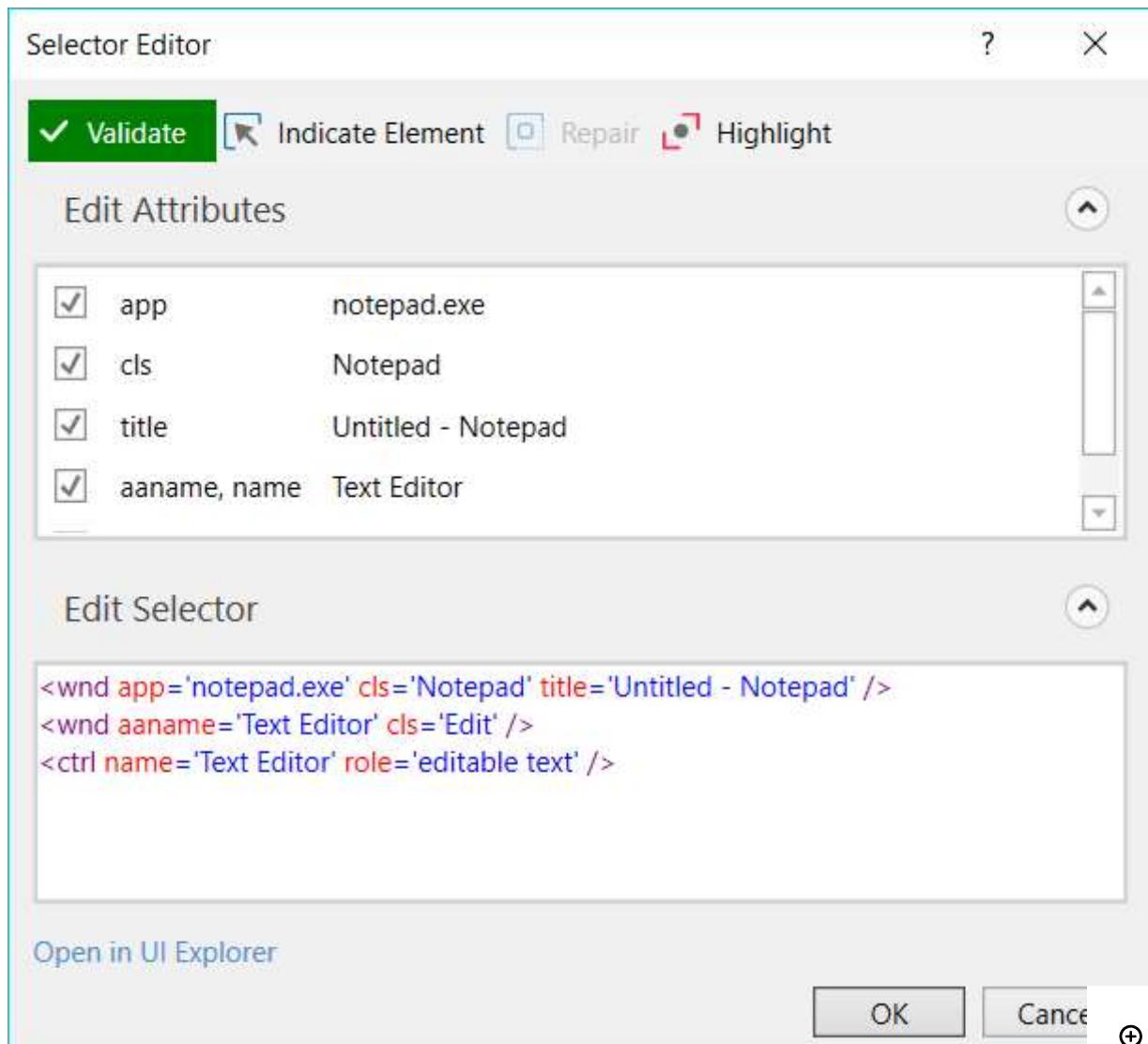
## Example of Generating a Selector With Wildcards in the Selector Editor Window

Part of the name of a Notepad window changes according to the .txt file you open with it. This is where a well-placed wildcard can really help. Do the following to generate it:

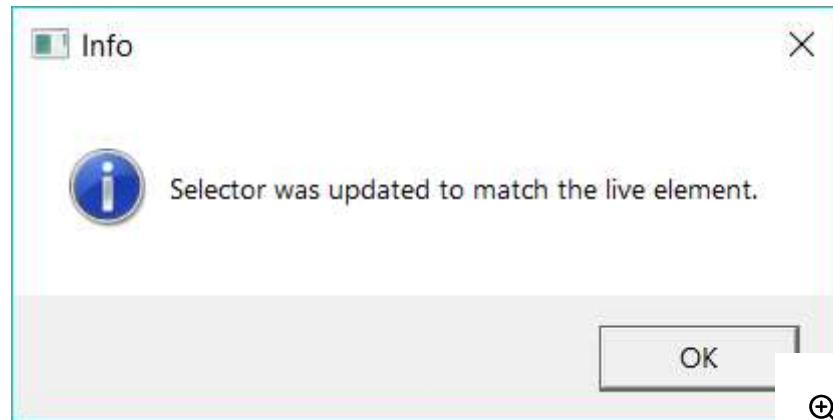
1. Open an empty Notepad window. Note that the window title is Untitled – Notepad.
2. In Studio, create a new sequence.
3. Drag a **Type Into** activity to the **Designer** panel.
4. Click **Indicate on Screen** and indicate the editable text field in Notepad. A selector is automatically generated and stored in the **Selector** field.



5. In the **Properties** panel, click the Ellipsis  button next to the **Selector** field. The **Selector Editor** window is displayed.



6. Open any .txt file with Notepad. Note that the window title is partially different than the one at step 1.
7. In Studio, in the **Selector Editor** window, click **Repair** and indicate the editable text field in Notepad window opened at step 6. A dialog box indicating that the selector was updated is displayed.



8. Click **OK**. The **Selector Editor** window and the selector are updated with a wildcard.

The screenshot shows the "Selector Editor" window. At the top, there are tabs: "Validate" (which is selected), "Indicate Element", "Repair", and "Highlight". Below the tabs, the "Edit Attributes" section displays the following list:

<input checked="" type="checkbox"/>	app	notepad.exe
<input checked="" type="checkbox"/>	cls	Notepad
<input checked="" type="checkbox"/>	title	* - Notepad
<input checked="" type="checkbox"/>	aaname, name	Text Editor

The "title" row has a red box around it, indicating it is the selected item. In the "Edit Selector" section below, the XML code is shown:

```
<wnd app='notepad.exe' cls='Notepad' title='* - Notepad' />
<wnd aaname='Text Editor' cls='Edit' />
<ctrl name='Text Editor' role='editable text' />
```

At the bottom of the window, there are buttons for "OK" and "Cancel", and a magnifying glass icon for search.



Default Theme

English

# Studio User Guide

RELEASE:

2023.4



## TABLE OF CONTENTS

### [Global Exception Handler](#)

[Handling Errors During Debugging](#)

[Example of Using the Global Exception Handler](#)

[Creating the Workflow](#)

[Adding a Global Exception Handler](#)

# Global Exception Handler

The **Global Exception Handler** is a type of workflow designed to determine the project's behavior when encountering an execution error. Only one **Global Exception Handler** can be set per automation project.

 **NOTE:**

The **Global Exception Handler** is not available for library projects, only processes.

The **Global Exception Handler** has two [arguments](#), that should **not** be removed.

The first argument is `errorInfo` with the **In** direction and it stores information about the error that was thrown and the workflow that failed. The level of the error to be logged can be set in the [Log Message](#) activity.

**ⓘ NOTE:**

Use the `ActivityInfo` property for `errorInfo` to get the name of the activity which threw the exception and view it in the **Output** panel.

The second argument, `result` has the **Out** direction and it is used for determining the next behavior of the process when it encounters an error. The following values can be assigned to the `result` argument:

- **Continue** - The exception is re-thrown.
- **Ignore** - The exception is ignored, and the execution continues from the next activity.
- **Retry** - The activity which threw the exception is retried. Use the `RetryCount` method for `errorInfo` to count the number of times the activity is retried.
- **Abort** - The execution stops after running the current **Global Exception Handler**.

**ⓘ NOTE:**

Any workflow may be flagged as a **Global Exception Handler** in Studio, except for library projects.

To control the workflow's behavior in case of an error, the **Global Exception Handler** retries the activity three times and then aborts with an error message.

## Handling Errors During Debugging

When an exception is detected during debugging, the activity which faulted is highlighted, the execution is paused, and the exception's type and details are mentioned in the **Locals** and **Call Stack** panels.

[Debugging actions](#) like **Continue**, **Stop**, **Retry**, **Ignore**, **Restart** and **Slow Step** are available in the ribbon. **Ignore** is used for continuing the execution from the next activity.

The **Retry** button retries to execute the current activity, without the Global Exception Handler stepping in. The **Continue** action runs the Global Exception Handler, taking into consideration the previously chosen values for the `result` argument, either **Continue**, **Ignore**, **Retry** or **Abort**.

**ⓘ NOTE:**

When using the **Global Exception Handler** with a project that includes a [Try Catch](#), make sure to group activities into a **Sequence** inside the **Try** container. Otherwise, the **Global Exception**

**Handler** does not execute.

In the case of nested activities, the **Global Exception Handler** executes for each activity in the call stack. However, it does not execute for activities directly encapsulated in a **Try Catch**, unless they're contained in an activity.

# Example of Using the Global Exception Handler

The following example showcases the project's behavior when an exception is thrown during execution.

The automation project is set to type some text in a TXT file and then close the application, but not before saving the file.

## Creating the Workflow

1. Create a **Blank Process** by following the steps in the [Creating a Basic Process](#) page.
2. Open Notepad and save a document on your machine. You can name the file `1.txt`.
3. In the Activities panel, search for [Use Application/Browser](#) and drag it to the Designer panel.
4. In Use Application/Browser:
  - Click **Indicate application to automate**, and then move the mouse pointer to the Notepad window. When the window is highlighted, click anywhere inside it.

The Use Application/Browser activity is updated, the path is added to the **Application path** field, and a screenshot of the window appears inside the activity.

- In the **Properties** panel, select the **Always** option for the **Close** property. This ensures Notepad is closed after the automation runs.
5. Add a [Type Into](#) activity in the **Use Application/Browser** activity's **Do** container. Click **Indicate in App** to select the Notepad window, and add enter a text between quotation marks in the **Type this** field. This activity writes the text into Notepad.
  6. From the **Activities** panel, add a [Keyboard Shortcuts](#) activity to the workflow. Indicate the Notepad window, then select **Record shortcut** and press **Ctrl + S** to record the key combination that saves the file after the text was typed in.

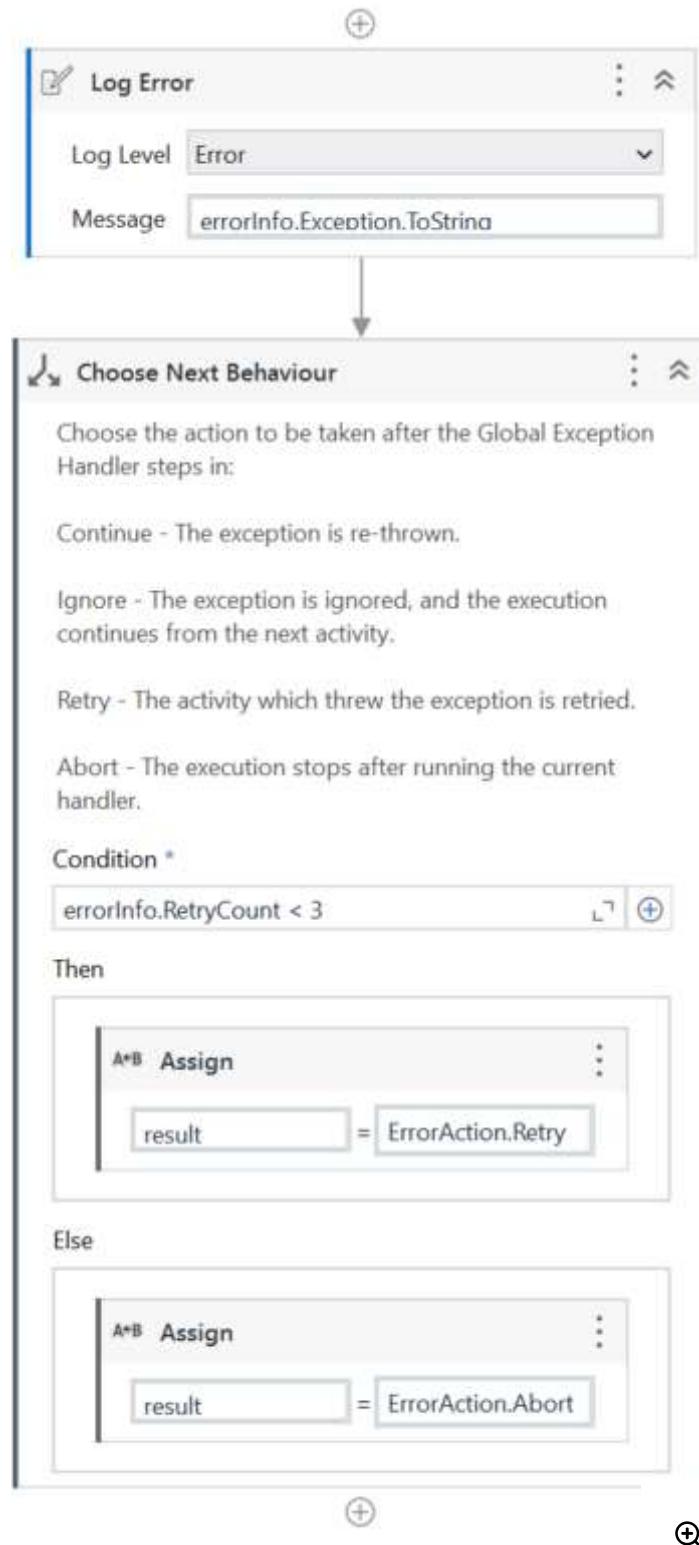
The resulted workflow should look like this:

The screenshot displays three stacked configuration cards within the Studio - Global Exception Handler interface:

- Top Card: Use Application: 1.txt - Notepad**
  - Icon: Preview image not available
  - Application path: "C:\Windows\System32\notepad.exe"
  - Application arguments: Text must be quoted
  - Match exact title: 1.txt - Notepad
- Middle Card: Type Into**
  - Icon: Indicate in App: 1.txt - Notepad
  - Text input field: "Typing something interesting"
  - Type this: Standard (radio button selected)
  - Empty field before typing: Single line [End, Shift+H]
  - Click before typing: Single
- Bottom Card: Keyboard Shortcuts**
  - Icon: App: 1.txt - Notepad
  - Record shortcut: Record shortcut (radio button selected)
  - Send key combination: Ctrl+S

# Adding a Global Exception Handler

1. In the **Design** tab part of the **Ribbon**, select **New > Global Handler**. The **New Global Handler** window opens. Type in a **Name** for the handler and save it in the project path. Click **Create**, a **Global Exception Handler** is added to the automation project.



2. Go back to the workflow you created earlier and modify it so that an activity fails to execute. For example, in the Use Application/Browser activity, select the **Match exact title** option and make sure the file is closed before you click **Run File** in the ribbon.

When the **Global Exception Handler** encounters an exception, it logs the name of the activity which faulted and starts retrying the activity three times. If it encounters the same exception each time and the number of retries reaches 3, the execution is aborted at the level of the activity which threw the exception.

If during one of the retries an exception isn't encountered, the execution of the workflow continues and the **Global Exception Handler** doesn't step in.

[Download example](#)



Default Theme

English

# UI Automation Activities

## TABLE OF CONTENTS

### [On Element Appear](#)

#### Properties

## On Element Appear

`UiPath.Core.Activities.OnUiElementAppear`

A container that waits for a UI element to appear and enables you to perform multiple actions within it.

## Properties

### Options

- **WaitVisible** - When this check box is selected, the activity waits for the specified UI element to be visible.
- **WaitActive** - When this check box is selected, the activity also waits for the specified UI element to be active.
- **RepeatForever** - Enables you to perpetually repeat this activity. Only boolean values (True, False) are supported.

## Output

- **FoundElement** - The resulted UI element. This field supports only `UiElement` variables.

## Common

- **DisplayName** - The display name of the activity.
- **ContinueOnError** - Specifies if the automation should continue even when the activity throws an error. This field only supports Boolean values (True, False). The default value is False. As a result, if the field is blank and an error is thrown, the execution of the project stops. If the value is set to True, the execution of the project continues regardless of any error.

 **NOTE:**

If this activity is included in **Try Catch** and the value of the **ContinueOnError** property is True, no error is caught when the project is executed.

## Misc

- **Private** - If selected, the values of variables and arguments are no longer logged at Verbose level.
- **Target.Selector** - Text property used to find a particular UI element when the activity is executed. It is actually a XML fragment specifying attributes of the GUI element you are looking for and of some of its parents.
- **Target.TimeoutMS** - Specifies the amount of time (in milliseconds) to wait for the activity to run before the `SelectorNotFoundException` error is thrown. The default value is 30000 milliseconds (30 seconds).
- **Target.WaitForReady** - Before performing the actions, wait for the target to become ready. The following options are available:
  - **None** - Does not wait for anything except the target UI element to exist before executing the action. For example, you can use this option if you want to retrieve just text from a web page or click a particular button, without having to wait for all UI elements to load. Note that this may have unwanted consequences if the button relies on elements which are not yet loaded, such as scripts.
  - **Interactive/Complete** - Waits all of the UI elements in the target app to exist before actually executing the action.

To assess if an application is in the Interactive or Complete state, the following tags are verified:

- **Desktop applications** - A `wm_null` message is sent to check the existence of the `<wnd>`, `<ctrl>`, `<java>`, or `<uia>` tags. If they exist, the activity is executed.
- **Web applications**:

1. **Internet Explorer** - The <webctrl> tag is used to check if the **Ready** state of the HTML document is set to **Complete**. Additionally, the **Busy** state has to be set to "False".

2. **Others** - The <webctrl> tag is used to check if the **Ready** state of the HTML document is **Complete**.

- **SAP applications** - First the presence of the <wnd> tag verified, after which a SAP specific API is used to detect if the session is busy or not.
- **Target.Element** - Use the `UiElement` variable returned by another activity. This property cannot be used alongside the `Selector` property. This field supports only `UiElement` variables.
- **Target.ClippingRegion** - Defines the clipping rectangle, in pixels, relative to the `UiElement`, in the following directions: left, top, right, bottom. It supports both positive and negative numbers.



Default Theme

English



# UI Automation Activities

## TABLE OF CONTENTS

### [On Element Vanish](#)

#### Properties

## On Element Vanish

`UiPath.Core.Activities.OnUiElementVanish`

A container that enables you to perform one or multiple actions after a specified UI element vanishes.

## Properties

### Input

- **WaitNotVisible** - When this check box is selected, the activity waits for the specified UI element to no longer be visible on the screen before performing any other action.
- **WaitNotActive** - When this check box is selected, the activity waits for the specified UI element to no longer be active before performing any other action.

### Common

- **DisplayName** - The display name of the activity.
- **ContinueOnError** - Specifies if the automation should continue even when the activity throws an error. This field only supports Boolean values (True, False). The default value is False. As a result, if the field is blank and an error is thrown, the execution of the project stops. If the value is set to True, the execution of the project continues regardless of any error.

 **NOTE:**

If this activity is included in **Try Catch** and the value of the **ContinueOnError** property is True, no error is caught when the project is executed.

## Options

- **RepeatForever** - Enables you to perpetually repeat this activity. Only boolean values (True, False) are supported.

## Misc

- **Private** - If selected, the values of variables and arguments are no longer logged at Verbose level.
- **Target.Selector** - Text property used to find a particular UI element when the activity is executed. It is actually a XML fragment specifying attributes of the GUI element you are looking for and of some of its parents.
- **Target.TimeoutMS** - Specifies the amount of time (in milliseconds) to wait for the activity to run before the SelectorNotFoundException error is thrown. The default value is 30000 milliseconds (30 seconds).
- **Target.WaitForReady** - Before performing the actions, wait for the target to become ready. The following options are available:
  - **None** - Does not wait for anything except the target UI element to exist before executing the action. For example, you can use this option if you want to retrieve just text from a web page or click a particular button, without having to wait for all UI elements to load. Note that this may have unwanted consequences if the button relies on elements which are not yet loaded, such as scripts.
  - **Interactive/Complete** - Waits all of the UI elements in the target app to exist before actually executing the action.

To assess if an application is in the Interactive or Complete state, the following tags are verified:

- **Desktop applications** - A `wm_null` message is sent to check the existence of the `<wnd>`, `<ctrl>`, `<java>`, or `<uia>` tags. If they exist, the activity is executed.
- **Web applications:**

1. **Internet Explorer** - The <webctrl> tag is used to check if the **Ready** state of the HTML document is set to **Complete**. Additionally, the **Busy** state has to be set to "False".
  2. **Others** - The <webctrl> tag is used to check if the **Ready** state of the HTML document is **Complete**.
- **SAP applications** - First the presence of the <wnd> tag verified, after which a SAP specific API is used to detect if the session is busy or not.
  - **Target.Element** - Use the `UiElement` variable returned by another activity. This property cannot be used alongside the `Selector` property. This field supports only `UiElement` variables.
  - **Target.ClippingRegion** - Defines the clipping rectangle, in pixels, relative to the `UiElement`, in the following directions: left, top, right, bottom. It supports both positive and negative numbers.



Default Theme

English



# UI Automation Activities

## TABLE OF CONTENTS

### [Extract Structured Data](#)

Properties

## Extract Structured Data

`UiPath.Core.Activities.ExtractData`

Extracts data from an indicated web page. You can specify what information to extract by providing an XML string in the **ExtractMetadata** field, in the Properties panel. This can easily be generated with all the properties set by using the Data Scraping wizard.

## Properties

### Input

- **ExtractMetadata** - An XML string that enables you to define what data to extract from the indicated web page.
- **Target.Selector** - Text property used to find a particular UI element when the activity is executed. It is actually a XML fragment specifying attributes of the GUI element you are looking for and of some of its parents.

- **Target.TimeoutMS** - Specifies the amount of time (in milliseconds) to wait for the activity to run before the SelectorNotFoundException error is thrown. The default value is 30000 milliseconds (30 seconds).
- **Target.WaitForReady** - Before performing the actions, wait for the target to become ready. The following options are available:
  - **None** - Does not wait for anything except the target UI element to exist before executing the action. For example, you can use this option if you want to retrieve just text from a web page or click a particular button, without having to wait for all UI elements to load. Note that this may have unwanted consequences if the button relies on elements which are not yet loaded, such as scripts.
  - **Interactive/Complete** - Waits all of the UI elements in the target app to exist before actually executing the action.

To assess if an application is in the Interactive or Complete state, the following tags are verified:

- **Desktop applications** - A `wm_null` message is sent to check the existence of the `<wnd>`, `<ctrl>`, `<java>`, or `<uia>` tags. If they exist, the activity is executed.
- **Web applications:**

1. **Internet Explorer** - The `<webctrl>` tag is used to check if the **Ready** state of the HTML document is set to **Complete**. Additionally, the **Busy** state has to be set to "False".
  2. **Others** - The `<webctrl>` tag is used to check if the **Ready** state of the HTML document is **Complete**.
- **SAP applications** - First the presence of the `<wnd>` tag verified, after which a SAP specific API is used to detect if the session is busy or not.
  - **Target.Element** - Use the `UiElement` variable returned by another activity. This property cannot be used alongside the `Selector` property. This field supports only `UiElement` variables.
  - **Target.ClippingRegion** - Defines the clipping rectangle, in pixels, relative to the `UiElement`, in the following directions: left, top, right, bottom. It supports both positive and negative numbers.

## Options

- **DelayBetweenPagesMS** - The amount of time, in milliseconds, to wait until the next page is loaded. (If the loading time of the page is longer, this value should be higher.)
- **MaxNumberOfResults** - The maximum number of results to be extracted. If the value is 0 all the identified elements are added to the output. The default value is 100.
- **NextLinkSelector** - The selector that identifies the link/button used to navigate to the next page. Should be relative to the `ExistingUiElement` property.
- **SendWindowMessages** - If selected, in the case where the data that is to be extracted spans multiple pages, the click that changes the page is executed by sending a specific message to the target application. This input method can work in the background, is compatible with most

desktop apps, but it is not the fastest of the methods. By default, this check box is not selected. If neither this nor the SimulateClick check boxes are selected, the default method simulates the click by using the hardware driver. The default method is the slowest, it cannot work in the background, but it is compatible with all desktop apps.

- **SimulateClick** - If selected, in the case where the data that is to be extracted spans multiple pages, it simulates the click that changes the page by using the technology of the target application. This input method is the fastest of the three and works in the background. By default, this check box is not selected. If neither this nor the SendWindowMessages check boxes are selected, the default method performs the click using the hardware driver. The default method is the slowest, it cannot work in the background, but it is compatible with all desktop apps.

## Output

- **DataTable** - The information extracted from the indicated web page. This field supports only DataTable variables.

## Common

- **DisplayName** - The display name of the activity.
- **ContinueOnError** - Specifies if the automation should continue even when the activity throws an error. This field only supports Boolean values (True, False). The default value is False. As a result, if the field is blank and an error is thrown, the execution of the project stops. If the value is set to True, the execution of the project continues regardless of any error.

### NOTE:

If this activity is included in **Try Catch** and the value of the **ContinueOnError** property is True, no error is caught when the project is executed.

## Misc

- **Private** - If selected, the values of variables and arguments are no longer logged at Verbose level.



Default Theme

English

# Workflow Activities

## TABLE OF CONTENTS

### [Trigger Scope](#)

Interface

Properties

## Trigger Scope

`UiPath.Core.Activities.TriggerScope`

Define actions that are triggered by various events.

## Interface

The activity interface is composed of the **Triggers** and **Actions** containers. This allows to define a set of triggers in the first, followed by a sequence of activities to execute based on trigger response.

The **Triggers** container only supports User Events activities.

In the **Actions** container, you can define **TriggerArgs** to be used in the next activity sequence. Please note that, in order to interrupt the Trigger Scope activity handling sequence, the **Break** activity should be used. This allows the workflow to continue execution as intended.

# Properties

## Common

- **ContinueOnError** - Specifies if the automation should continue even when the activity throws an error. This field only supports Boolean values (True, False). The default value is False. As a result, if the field is blank and an error is thrown, the execution of the project stops. If the value is set to True, the execution of the project continues regardless of any error.

 **NOTE:**

If this activity is included in **Try Catch** and the value of the **ContinueOnError** property is True, no error is caught when the project is executed.

- **DisplayName** - The display name of the activity.

## Misc

- **Private** - If selected, the values of variables and arguments are no longer logged at Verbose level.

## Options

- **SchedulingMode.Sequential** - Actions are executed one after another.
- **SchedulingMode.Concurrent** - Action executions can overlap.
- **SchedulingMode.OneTime** - Execute one action and exit.

 **NOTE:**

If a trigger is received while a process is paused, it fires after the process resumes.



Default Theme

English

# Studio User Guide

RELEASE: 2023.4 ▾

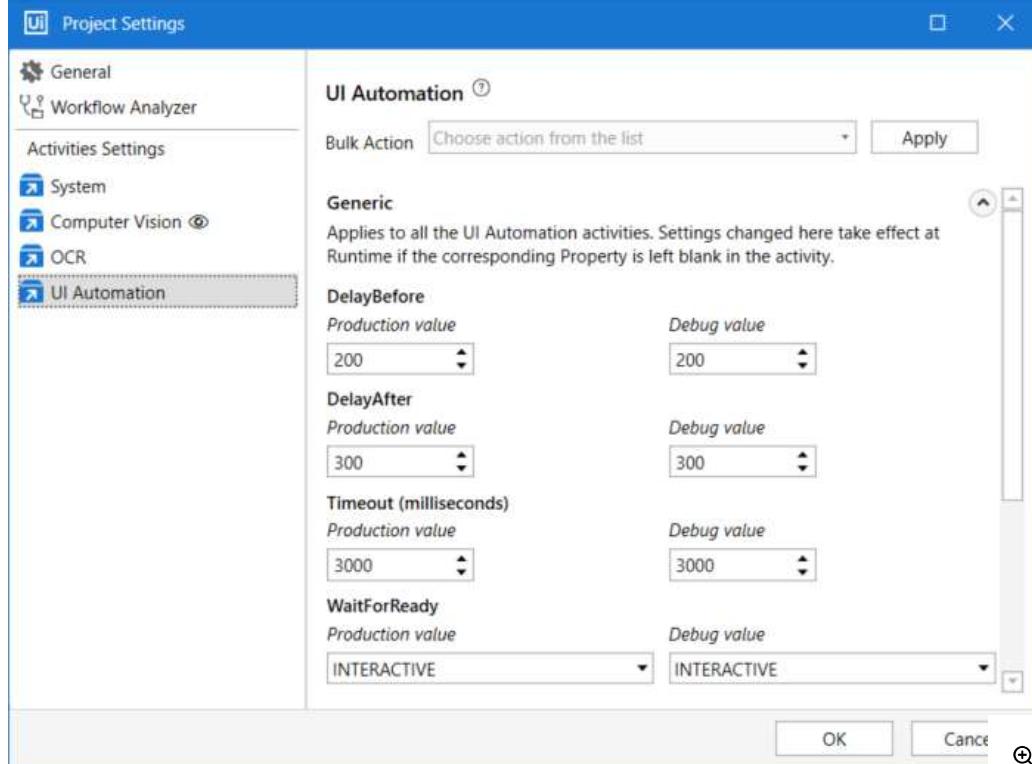
## TABLE OF CONTENTS

### [Configuring Activity Project Settings](#)

## Configuring Activity Project Settings

**Activity Project Settings** represent a set of changes that can be configured at the project level and applied to all activities part of project dependencies.

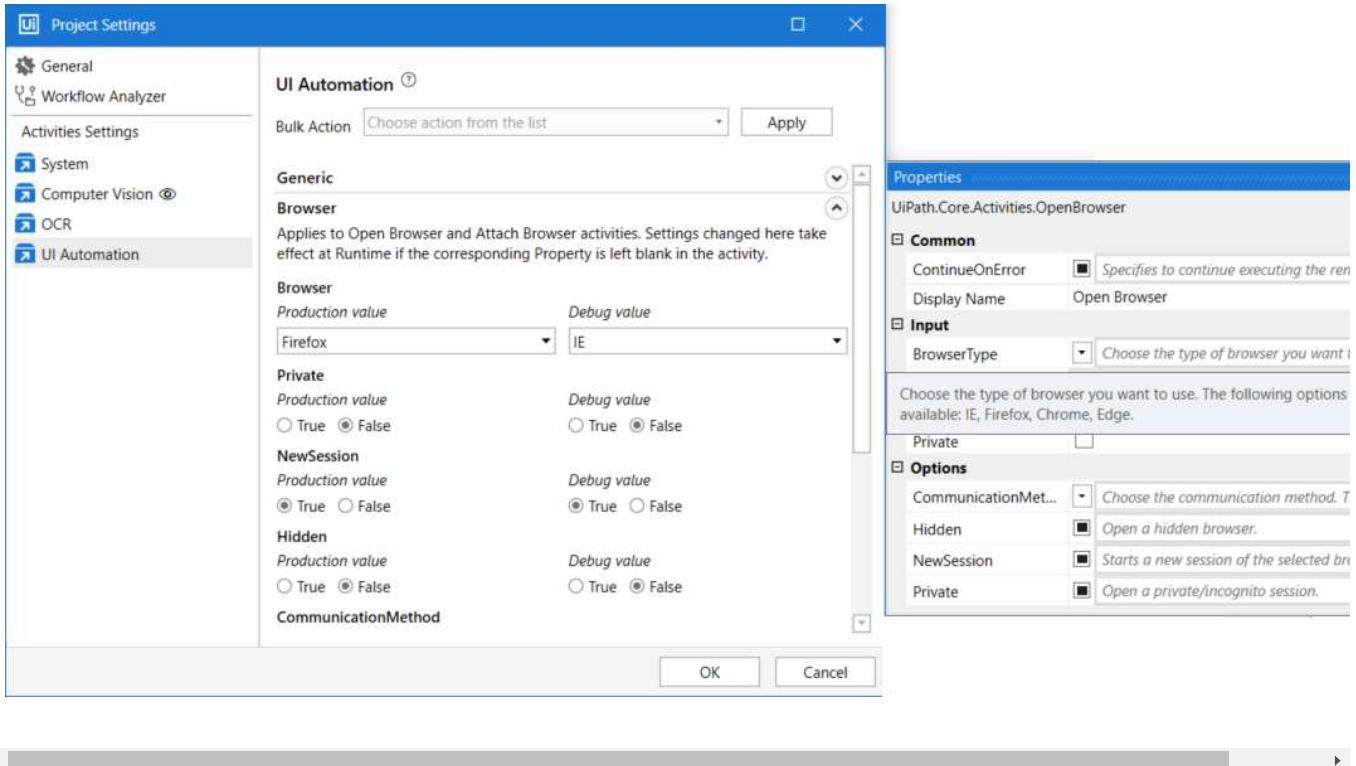
To configure activity properties at project level click the  icon in the **Project** panel. The **Project Settings** window opens:



Tabs under **Activity Settings** contain the settings available for each installed activity package and for some bundled dependencies like **Computer Vision**, part of **UIAutomation**.

These settings can be configured for two scenarios: **Run** and **Debug**. In this way, for the same activities in the project, specific settings are applied when the project is run, while a different set of settings are applied during debugging.

For example, the **Open Browser** activity contains the **BrowserType** property with the **IE** default browser for opening the URL. If in **Activity Project Settings**, the same property is set to open URLs in Firefox, then this setting overwrites defaults and applies to all activities with this property used in the project.



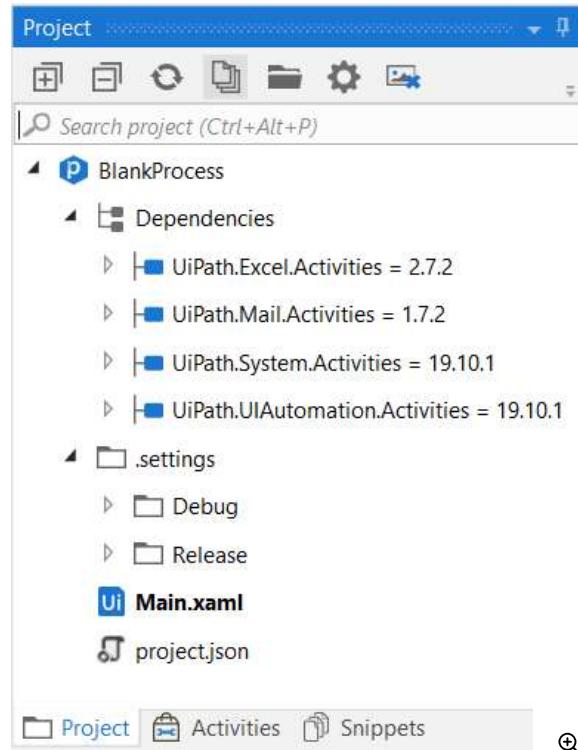
If in the same project one activity has the **BrowserType** property set to **Chrome** in its **Properties** panel, then the value set in the panel overwrites any other value configured in the **Activity Project Settings** window.

A set of actions are available in the **Bulk Action** drop-down menu for performing changes to multiple settings:

- **Reset all**
- **Reset run settings**
- **Reset debug settings**
- **Copy debug settings to run**
- **Copy run settings to debug**.

Please bear in mind that even though activity properties are changed from the **Property Settings** window and new values are applied throughout the project, the default values in the **Properties** panel are not visibly updated. The change is applied, but not visible in the **Properties** panel.

The configured activity settings are saved in the project location, and are visible in the **Project** panel, under the **.settings** folder.



**NOTE:**

Activity Project Settings are not taken into account during runtime or debugging if the **.settings** folder is renamed or deleted.

To save your **Activity Project Settings** configuration and apply it to other future automation projects use the [Save as Template](#) functionality in Studio.

To find out more about the settings available for various package, Project Settings pages are available in the Activities guide for the packages that support this feature, for example, [Project Settings for UI Automation Modern](#).



Default Theme

English



# Workflow Activities

## TABLE OF CONTENTS

### [Project Settings](#)

[Overview](#)

[Configuring Testing Project Settings](#)

[Keep Screenshots](#)

[Screenshots Path](#)

[Verify Activities Output Format](#)

# Project Settings

## Overview

Configure changes at the project level. These global settings will impact several or all testing activities. For more information see [Configuring Activity Project Settings](#).

## Configuring Testing Project Settings

1. In the **Project** panel, go to **Project Settings**.
2. Navigate to **Activities Settings > Testing**.
3. Choose the settings you want to configure:

- [Keep Screenshots](#)
- [Screenshots Path](#)
- [Verify Activities Output Format](#)

4. Click **OK** to save changes.

You can reset all settings to their default values or copy all values between the debug and production fields using the **Bulk Actions** drop-down menu at the top of the window.

## Keep Screenshots

Property Name	Description	Impacted Activities	Default Setting
<b>Run value</b>	Keep screenshots to your production environment.	N/A	None
<b>Debug value</b>	Keep screenshots to your testing environment.	N/A	None

## Screenshots Path

Property Name	Description	Impacted Activities	Default Setting
<b>Run value</b>	Set default storage location for your production environment.	N/A	None
<b>Debug value</b>	Set default storage location for your testing environment.	N/A	None

## Verify Activities Output Format

You can configure the format of the output message globally, for your verification activities:

- [Verify Control Attribute](#)
- [Verify Expression](#)
- [Verify Expression with Operator](#)
- [Verify Range](#)

Property Name	Description	Impacted Activities	Default Setting
<b>Run value</b>	Specify the format of the output message for your production environment.	Verify Control Attribute Verify Expression Verify Expression with Operator Verify Range	None
<b>Debug value</b>	Specify the format of the output message for your testing environment.	Verify Control Attribute Verify Expression Verify Expression with Operator Verify Range	None



Default Theme

English



# Document Understanding Activities

## TABLE OF CONTENTS

[Extract PDF Page Range](#)

Properties

## Extract PDF Page Range

`UiPath.PDF.Activities.PDF.ExtractPDFPageRange`

Extracts a specified range of pages from a PDF document.

## Properties

### Common

- **DisplayName** - The display name of the activity.

### File

- **FileName** - The path of the PDF file you want to extract a range of pages from. This field supports only strings and String variables.
- **OutputFileName** - The name you want to use for the file that is generated from the extracted range of pages. This field supports only strings and String variables.

- **Password** - The password of the PDF file, if necessary. This field supports only strings and String variables.

## Input

- **Range** - The range of pages that you want to retrieve. You can specify a single page (e.g. "7"), a range of pages (e.g. "7-12"), or a complex range, (e.g. "2-5, 7, 15-End" or "All"). Only string variables and strings are supported. By default, this field is cleared.

## Misc

- **Private** - If selected, the values of variables and arguments are no longer logged at Verbose level.



Default Theme

English



# Workflow Activities

## TABLE OF CONTENTS

---

### [Try Catch](#)

Description

Project compatibility

Configuration

Properties

Example of using the Try Catch activity

## Try Catch

`System.Activities.Statements.TryCatch`

## Description

Catches a specified exception type in a sequence or activity, and either displays an error notification or dismisses it and continues the execution.

There is no limit to how many **Catches** you can use in a **Try Catch** activity. This activity requires at least one catch to be added.

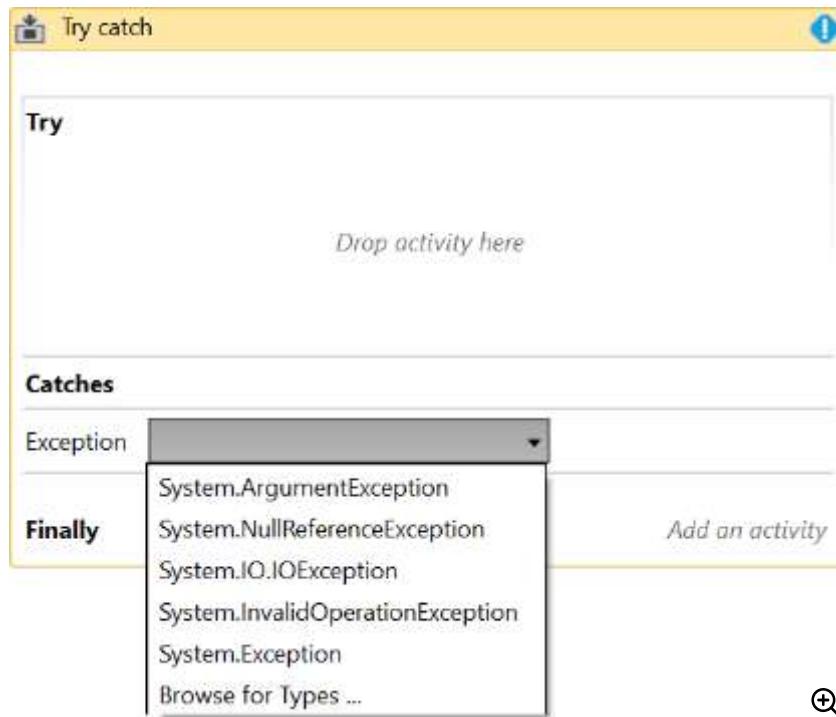
# Project compatibility

Windows - Legacy | Windows | Cross-platform

## Configuration

The activity body contains three fields:

- **Try** - The activity performed which has a chance of throwing an error.
- **Catches** - The activity or set of activities to be performed when an error occurs.
  - **Exception** - The exception type to look for. You can add multiple exceptions.
- **Finally** - The activity or set of activities to be performed after the **Try** and **Catches** blocks are executed. This section is executed only when no exceptions are thrown or when an error occurs and is caught in the **Catches** section.



**NOTE:**

- If an activity is included in the **Try** section, and the value of the **ContinueOnError** property is True, no error is caught when the project is executed.
- The **Try Catch** activity does not catch fatal exceptions such as:
  - **FatalException**
  - **OutOfMemoryException**
  - **ThreadAbortException**
  - **FatalInternalException**

# Properties

## Common

- **DisplayName** - The display name of the activity.

## Misc

- **Private** - If selected, the values of variables and arguments are no longer logged at Verbose level.

 **NOTE:**

Pressing “Ctrl + T” places the selected activity inside the **Try** section of a **Try Catch** activity.

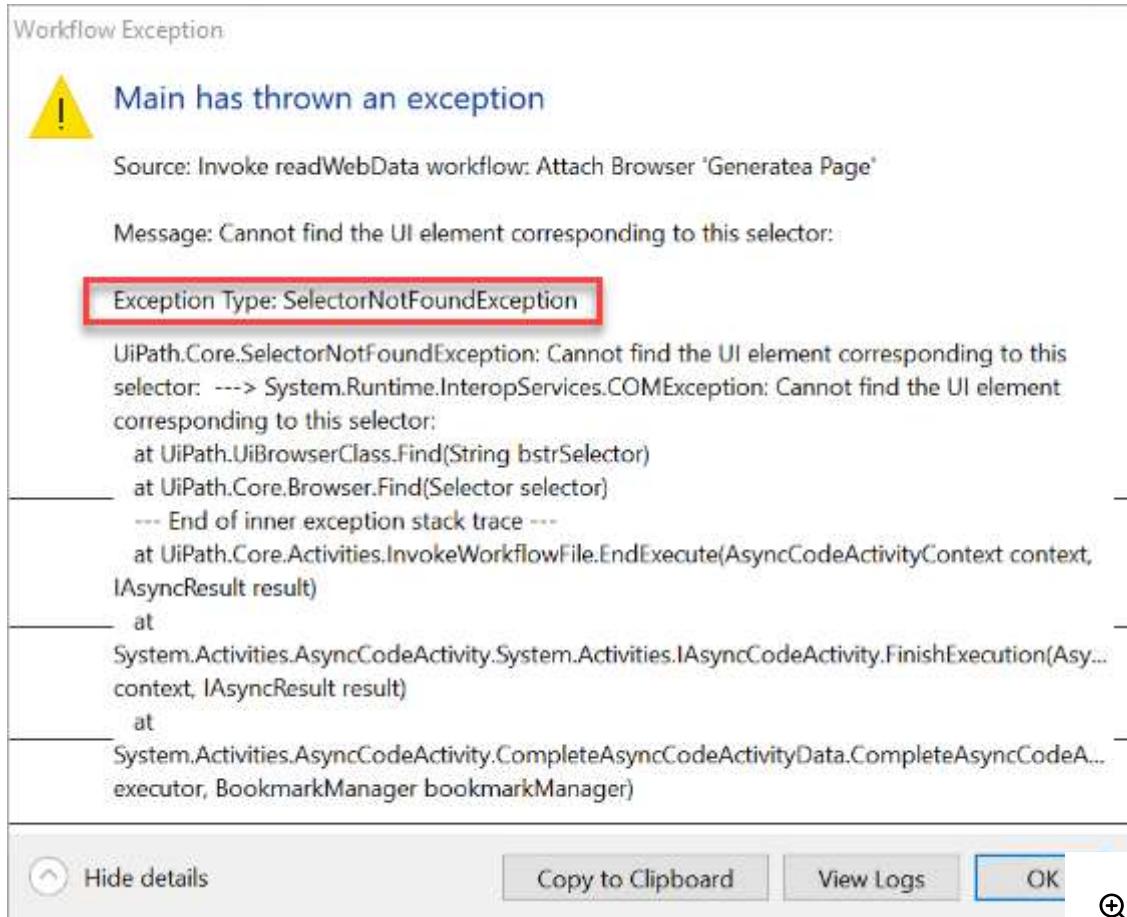
# Example of using the Try Catch activity

To better understand the importance of the **Try Catch** activity, we created an automation that gathers multiple names from a random name generator website and writes them in an Excel spreadsheet.

[You can download the initial workflow here.](#)

A **Build Data Table** activity is used to create a table in which to store the gathered names. Another workflow is invoked to read the web data. Finally, an **Excel application scope** activity is used to write the gathered information in the Excel file.

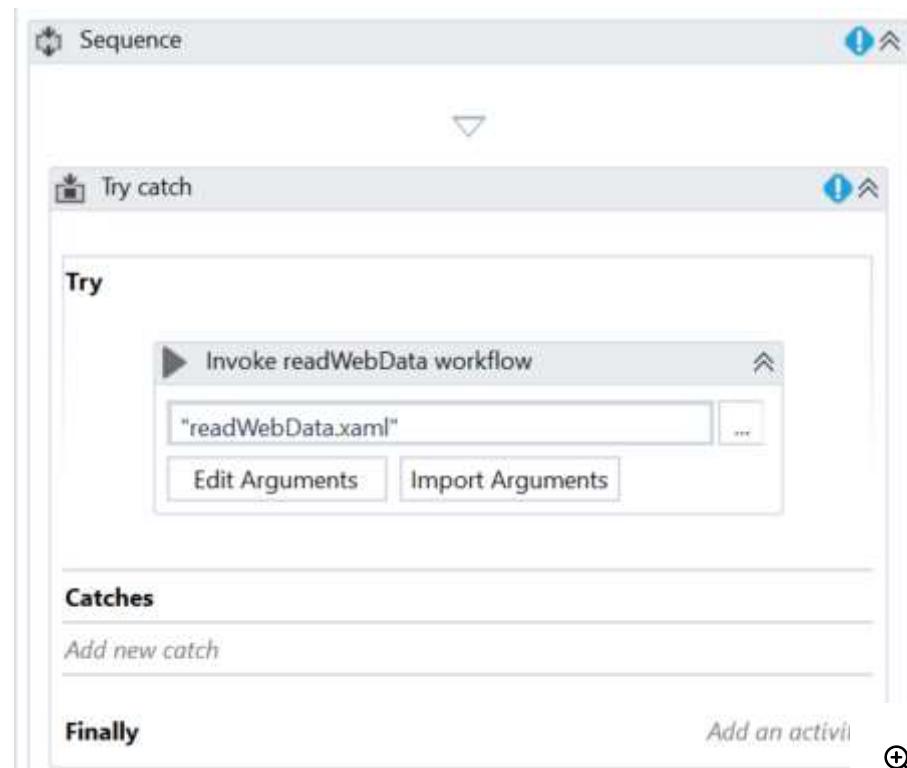
First of all, let's run the automation to check for any errors. Notice that a **Workflow Exception** window is displayed. The **Exception Type** field tells us what the problem is. This is used in the **Catches** section of a **Try Catch** as the exception type to look for during the workflow execution.



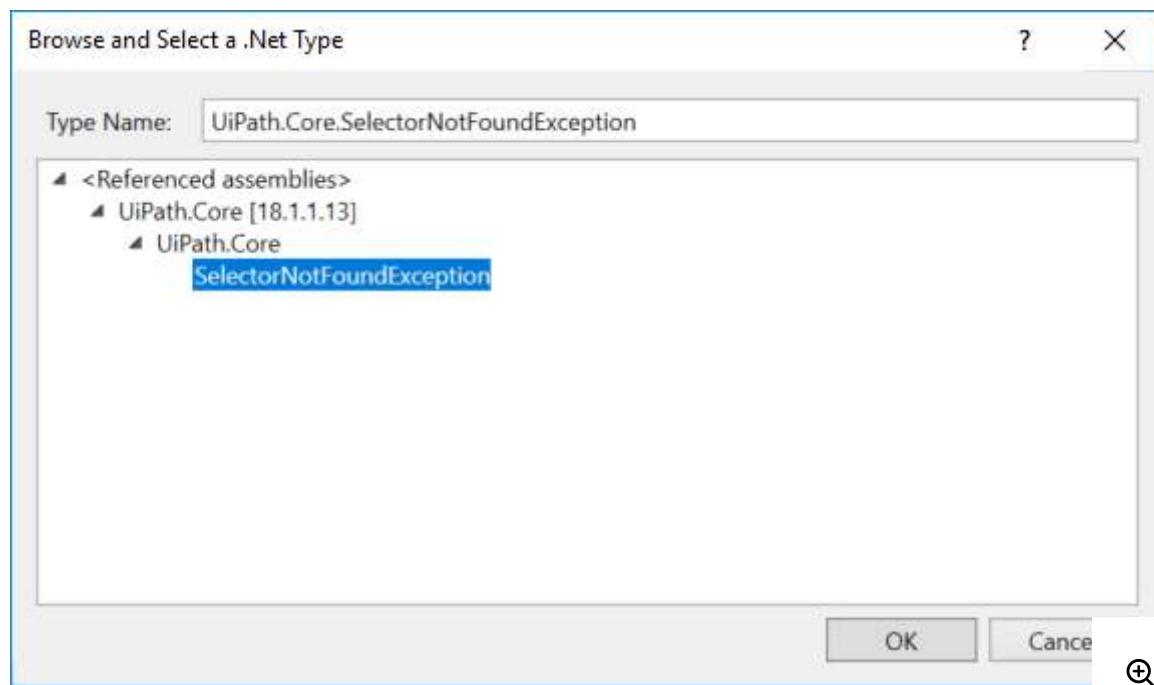
As you can see in the screenshot above, when running the example workflow, there seems to be a problem with the **Attach Browser** container selector. The issue is that the selector fails to identify the browser window with the "Generate a Random Name - Fake Name Generator" name.

To catch this exception, we need to perform the following:

1. Drag the **Try Catch** activity from the **Activities** panel above the **Invoke workflow** activity.
2. Place the **Invoke workflow** activity in the **Try** section of the **Try Catch** activity. This watches the **Invoke workflow** activity in case it throws an error.

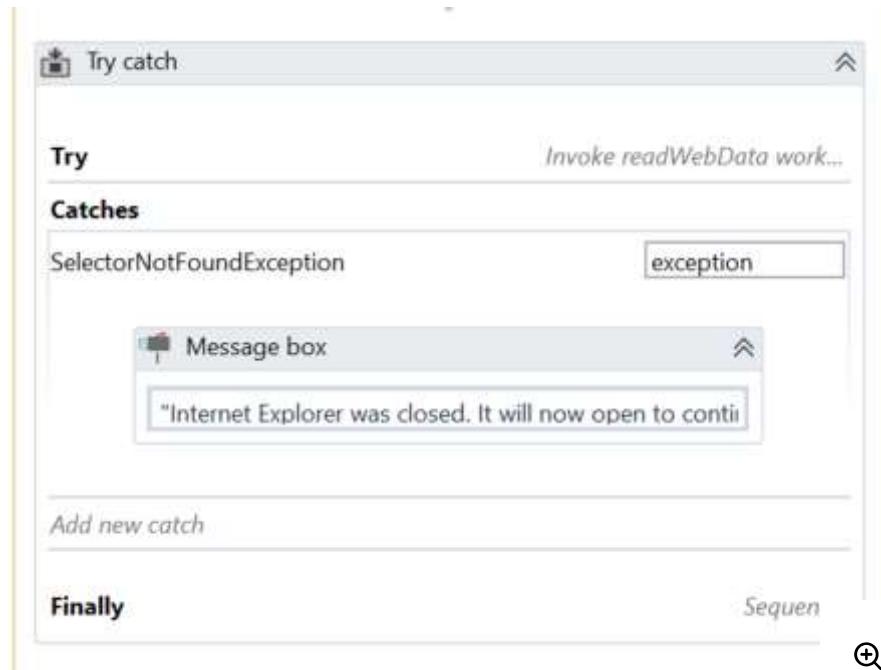


3. In the **Catches** section, select the `UiPath.Core.SelectorNotFoundException` exception from the drop-down. If it's not there, you can find it in the [Browse and Select a .Net Type](#) window.



4. Optionally, you can add a **Message Box** activity in the **Catches** section. You can fill in the **Content** field with an informative message between quotes, in our case "Internet Explorer was closed. It

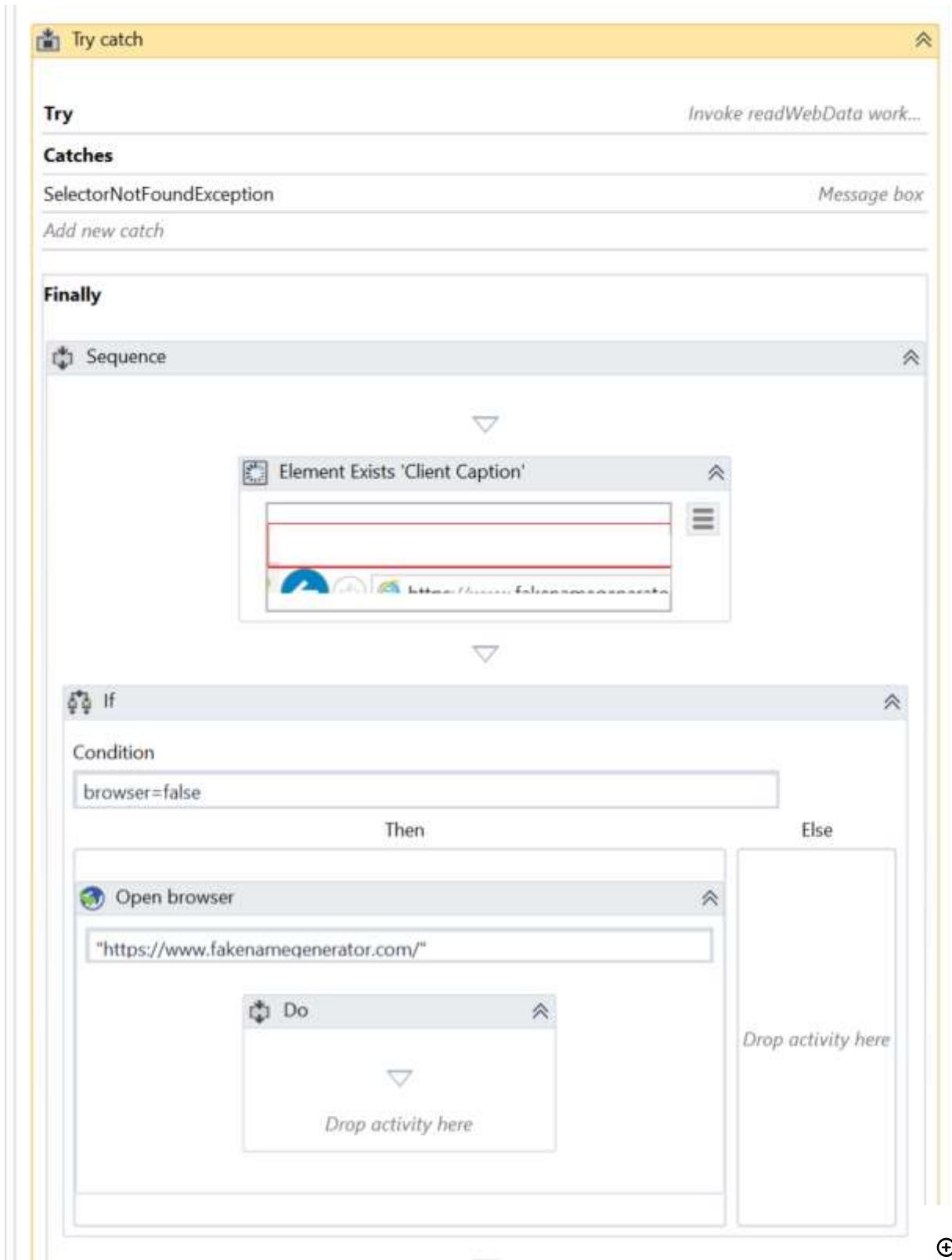
will now open to continue the workflow execution". This means that whenever the exception is caught, this message box is displayed, to inform the user that the browser is about to open so that the workflow is successfully executed.



5. Drag the **Element Exists** activity in the **Finally** section. This is used to check if Internet Explorer is open on the page of interest, <https://www.fakenamegenerator.com>.
6. Open Internet Explorer and access the previously mentioned page.
7. Use the **Indicate on screen** functionality to select the Internet Explorer window.
8. Select the **Element Exists** activity and edit its selector so that it looks like this
 

```
app='iexplore.exe' title='Generate a Random Name - Fake Name Generator - Internet Explorer' />
```

 This selector ensures that the **Element Exists** activity only looks for an active Internet Explorer window in which the aforementioned page is open.
9. In the **Output** property, create a variable with a relevant name, such as `browser`. This is a boolean variable which helps you determine whether or not Internet Explorer is active on the indicated page.
0. Add an **If** activity under the **Element Exists** activity. This is used to open Internet Explorer if it's closed, and continue the workflow otherwise.
  1. In the **Condition** field, write `browser=false`. This condition is used to verify if the browser is opened or not, and perform other actions, based on its value.
  2. Drag an **Open Browser** activity in the **Then** section. If the **Condition** is met (the browser is closed), then the **Open Browser** activity is used to open it, without affecting the workflow.
  3. In the **Url** field type <https://www.fakenamegenerator.com>.
  4. Leave the **Else** section empty so that the workflow continues as expected if Internet Explorer is already opened on the indicated website.



5. Run the workflow and notice one of the following:

- If Internet Explorer is closed - The user is informed that Internet Explorer is about to open so the workflow can continue. The browser opens, all expected data is gathered and written to the Excel file.

- If Internet Explorer is open - The workflow is executed as expected.

[Download example](#)



Default Theme

English



# Orchestrator Standalone User Guide

DELIVERY:

AUTOMATION CLOUD

AUTOMATION SUITE

STANDALONE

RELEASE:

2023.4

TABLE OF CONTENTS

[Business Exception Vs Application Exception](#)

## Business Exception Vs Application Exception

It is important to choose the correct type of exception with which a transaction is failed, because this choice influences whether **Orchestrator** chooses to retry the transaction of the queue item or not, as follows:

- An **Application Exception** describes an error rooted in a technical issue, such as an application that is not responding.

Such a situation is, for example, a project which extracts phone numbers from an employee database, creating queue items for each of them. These items are then to be processed and inserted into a financial application. If, when the transaction is attempted, the financial application freezes, the Robot cannot find the field where it should insert the phone number, and eventually throws an error.

These kinds of issues have a chance of being solved simply by retrying the transaction, as the application can unfreeze.

- A **Business Exception** describes an error rooted in the fact that certain data which the automation project depends on is incomplete or missing.

Such a situation is, for example, a project which extracts phone numbers from an employee database, creating queue items for each of them. These items are then to be processed and inserted into a financial application. If a certain phone number is missing a digit due to human error, the queue item containing it becomes invalid. This causes the automation to throw an exception, as the Phone Number field in the financial application does not accept a queue item that contains an incomplete number.

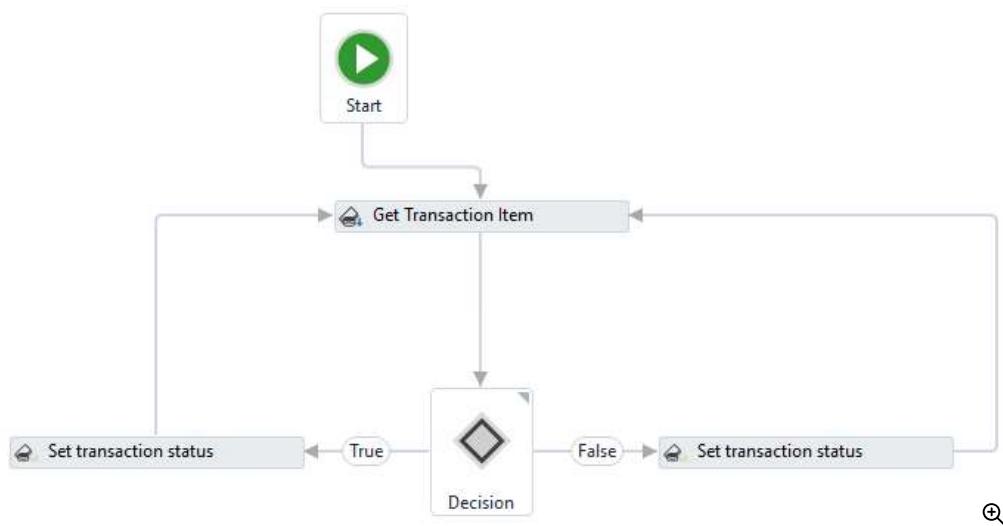
Retrying the transaction does not yield any chance of solving the issue, and there are other better courses of action, such as notifying the human user of this error.

The **Set Transaction Status** activity can be used to shape the logic of your project in a way that encapsulates this distinction in several ways:

- If the **Set Transaction Status** activity fails the transaction with an **Application Exception** and the **Auto Retry** option in the **Create Queue** page is set to **Yes** when the queue is created, the queue item is retried.
- By default, Orchestrator does **not** retry transactions which are failed due to **Business Exceptions**. This happens because an inconsistency between the transaction value and the business requirement means that there might be errors in the initial data which the queue items were created from. Additional actions might be required to fix this type of issue, and logging this type of exception can be useful.
- An **If** or **Flow Decision** activity can be used to take different courses of action if a transaction is failed with a certain type of exception, such as using the **Log Message** activity to log a custom message or the **Message Box** activity to display a window containing information

about the event.

Below you can see an example of such a project:



The screenshot below shows the mapping of the properties in the **Set Transaction Status** activity (on the left) and their corresponding fields in the **Transaction Details** window in Orchestrator.

Properties		Item Details for 1d1a17be-1526-4522-aa20-f77b2f2495b8																																		
UiPath.Core.Activities.SetTransactionStatus																																				
Common																																				
DisplayName	Set transaction status																																			
TimeoutMS	Specifies the amount of time (																																			
Input																																				
Output	(Collection)																																			
Status	<b>1</b> Failed																																			
TransactionItem	transItem																																			
Misc																																				
Private	<input type="checkbox"/>																																			
Transaction Error																																				
AssociatedFilePath	The full path of a file that is a...																																			
Details	Details regarding the failed Ti...																																			
ErrorType	<b>2</b> Business																																			
Reason	<b>3</b> "doesn't contain the letter F"																																			
<b>Item Details for 1d1a17be-1526-4522-aa20-f77b2f2495b8</b>																																				
<b>Exception</b> <ul style="list-style-type: none"> <li>Reason: doesn't contain the letter F <b>3</b></li> </ul>																																				
<table border="1"> <thead> <tr> <th>STATUS</th> <th>REVISION</th> <th>DEADLINE</th> <th>RETRYNO.</th> <th>POSTPONE</th> <th>STARTED</th> <th>ENDED</th> <th>ROBOT</th> <th>EX</th> </tr> </thead> <tbody> <tr> <td>Failed</td> <td>None</td> <td></td> <td>0</td> <td></td> <td>8 days ago</td> <td>8 days ago</td> <td>Documentat...</td> <td>Bu</td> </tr> <tr> <td><b>1</b></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>										STATUS	REVISION	DEADLINE	RETRYNO.	POSTPONE	STARTED	ENDED	ROBOT	EX	Failed	None		0		8 days ago	8 days ago	Documentat...	Bu	<b>1</b>								
STATUS	REVISION	DEADLINE	RETRYNO.	POSTPONE	STARTED	ENDED	ROBOT	EX																												
Failed	None		0		8 days ago	8 days ago	Documentat...	Bu																												
<b>1</b>																																				

The **True** branch of the **Flow Decision** activity sets the transaction status to Failed with a **Business Exception**, while the **False** branch sets it to Failed with an **Application Exception**.



Default Theme

English

# IOException Class

Reference

## Definition

Namespace: [System.IO](#)

Assembly: System.Runtime.dll

The exception that is thrown when an I/O error occurs.

C#

```
public class IOException : SystemException
```

Inheritance Object → Exception → SystemException → IOException

Derived [System.IO.DirectoryNotFoundException](#)

[System.IO.DriveNotFoundException](#)

[System.IO.EndOfStreamException](#)

[System.IO.FileLoadException](#)

[System.IO.FileNotFoundException](#)

[More...](#)

## Examples

This code example is part of a larger example provided for the [FileStream.Lock](#) method.

C#

```
// Catch the IOException generated if the
// specified part of the file is locked.
catch(IOException e)
{
    Console.WriteLine(
        "{0}: The write operation could not " +
        "be performed because the specified " +
        "part of the file is locked.",
        e.GetType().Name);
}
```

## Remarks

[IOException](#) is the base class for exceptions thrown while accessing information using streams, files and directories.

The Base Class Library includes the following types, each of which is a derived class of [IOException](#) :

- [DirectoryNotFoundException](#)
- [EndOfStreamException](#)
- [FileNotFoundException](#)
- [FileLoadException](#)
- [PathTooLongException](#)

Where appropriate, use these types instead of [IOException](#).

[IOException](#) uses the HRESULT COR\_E\_IO which has the value 0x80131620.

## Constructors

<a href="#">IOException()</a>	Initializes a new instance of the <a href="#">IOException</a> class with its message string set to the empty string (""), its HRESULT set to COR_E_IO, and its inner exception set to a null reference.
<a href="#">IOException(SerializationInfo, StreamingContext)</a>	Initializes a new instance of the <a href="#">IOException</a> class with the specified serialization and context information.
<a href="#">IOException(String)</a>	Initializes a new instance of the <a href="#">IOException</a> class with its message string set to <code>message</code> , its HRESULT set to COR_E_IO, and its inner exception set to <code>null</code> .
<a href="#">IOException(String, Exception)</a>	Initializes a new instance of the <a href="#">IOException</a> class with a specified error message and a reference to the inner exception that is the cause of this exception.
<a href="#">IOException(String, Int32)</a>	Initializes a new instance of the <a href="#">IOException</a> class with its message string set to <code>message</code> and its HRESULT user-defined.

## Properties

Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from <a href="#">Exception</a> )
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from <a href="#">Exception</a> )
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from <a href="#">Exception</a> )
InnerException	Gets the <a href="#">Exception</a> instance that caused the current exception. (Inherited from <a href="#">Exception</a> )
Message	Gets a message that describes the current exception. (Inherited from <a href="#">Exception</a> )
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from <a href="#">Exception</a> )
StackTrace	Gets a string representation of the immediate frames on the call stack. (Inherited from <a href="#">Exception</a> )
TargetSite	Gets the method that throws the current exception. (Inherited from <a href="#">Exception</a> )

## Methods

Equals(Object)	Determines whether the specified object is equal to the current object. (Inherited from <a href="#">Object</a> )
GetBaseException()	When overridden in a derived class, returns the <a href="#">Exception</a> that is the root cause of one or more subsequent exceptions. (Inherited from <a href="#">Exception</a> )
GetHashCode()	Serves as the default hash function. (Inherited from <a href="#">Object</a> )
GetObjectData(SerializationInfo, StreamingContext)	When overridden in a derived class, sets the <a href="#">SerializationInfo</a> with information about the exception. (Inherited from <a href="#">Exception</a> )
GetType()	Gets the runtime type of the current instance. (Inherited from <a href="#">Exception</a> )
MemberwiseClone()	Creates a shallow copy of the current <a href="#">Object</a> .

(Inherited from [Object](#))

<a href="#">ToString()</a>	Creates and returns a string representation of the current exception. (Inherited from <a href="#">Exception</a> )
----------------------------	--

## Events

<a href="#">SerializeObjectState</a>	<b>Obsolete.</b> Occurs when an exception is serialized to create an exception state object that contains serialized data about the exception. (Inherited from <a href="#">Exception</a> )
--------------------------------------	--

## Applies to

Product	Versions
.NET	Core 1.0, Core 1.1, Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 2.0, 2.1
UWP	10.0
Xamarin.iOS	10.8
Xamarin.Mac	3.0

## See also

- [Exception](#)
- [Handling and throwing exceptions in .NET](#)
- [File and Stream I/O](#)
- [How to: Read Text from a File](#)
- [How to: Write Text to a File](#)

# NullReferenceException Class

Reference

## Definition

Namespace: [System](#)

Assembly: System.Runtime.dll

The exception that is thrown when there is an attempt to dereference a null object reference.

C#

```
public class NullReferenceException : SystemException
```

Inheritance Object → Exception → SystemException → NullReferenceException

## Remarks

A [NullReferenceException](#) exception is thrown when you try to access a member on a type whose value is `null`. A [NullReferenceException](#) exception typically reflects developer error and is thrown in the following scenarios:

- You've forgotten to instantiate a reference type. In the following example, `names` is declared but never instantiated:

```
C#  
  
using System;  
using System.Collections.Generic;  
  
public class Example  
{  
    public static void Main(string[] args)  
    {  
        int value = Int32.Parse(args[0]);  
        List<String> names;  
        if (value > 0)  
            names = new List<String>();  
  
        names.Add("Major Major Major");  
    }  
}
```

```
// Compilation displays a warning like the following:  
//   Example1.cs(10) : warning BC42104: Variable //names// is used before it  
//   has been assigned a value. A null reference exception could result  
//   at runtime.  
//  
//           names.Add("Major Major Major")  
//           ~~~~~  
// The example displays output like the following output:  
// Unhandled Exception: System.NullReferenceException: Object reference  
//   not set to an instance of an object.  
//   at Example.Main()
```

Some compilers issue a warning when they compile this code. Others issue an error, and the compilation fails. To address this problem, instantiate the object so that its value is no longer `null`. The following example does this by calling a type's class constructor.

C#

```
using System;  
using System.Collections.Generic;  
  
public class Example  
{  
    public static void Main()  
    {  
        List<String> names = new List<String>();  
        names.Add("Major Major Major");  
    }  
}
```

- You've forgotten to dimension an array before initializing it. In the following example, `values` is declared to be an integer array, but the number of elements that it contains is never specified. The attempt to initialize its values therefore thrown a `NullReferenceException` exception.

C#

```
using System;  
  
public class Example  
{  
    public static void Main()  
    {  
        int[] values = null;  
        for (int ctr = 0; ctr <= 9; ctr++)
```

```
values[ctr] = ctr * 2;

    foreach (var value in values)
        Console.WriteLine(value);
    }
}
// The example displays the following output:
//     Unhandled Exception:
//         System.NullReferenceException: Object reference not set to an
//             instance of an object.
//             at Example.Main()
```

You can eliminate the exception by declaring the number of elements in the array before initializing it, as the following example does.

C#

```
using System;

public class Example
{
    public static void Main()
    {
        int[] values = new int[10];
        for (int ctr = 0; ctr <= 9; ctr++)
            values[ctr] = ctr * 2;

        foreach (var value in values)
            Console.WriteLine(value);
    }
}
// The example displays the following output:
//     0
//     2
//     4
//     6
//     8
//     10
//     12
//     14
//     16
//     18
```

For more information on declaring and initializing arrays, see [Arrays](#) and [Arrays](#).

- You get a `null` return value from a method, and then call a method on the returned type. This sometimes is the result of a documentation error; the documentation fails to note that a method call can return `null`. In other cases, your code erroneously assumes that the method will always return a non-`null` value.

The code in the following example assumes that the [Array.Find](#) method always returns `Person` object whose `FirstName` field matches a search string. Because there is no match, the runtime throws a [NullReferenceException](#) exception.

C#

```
using System;

public class Example
{
    public static void Main()
    {
        Person[] persons = Person.AddRange( new String[] { "Abigail",
"Abra",
"Ariella",
"Arnold", "Aston", "Astor" } );
        String nameToFind = "Robert";
        Person found = Array.Find(persons, p => p.FirstName ==
nameToFind);
        Console.WriteLine(found.FirstName);
    }
}

public class Person
{
    public static Person[] AddRange(String[] firstNames)
    {
        Person[] p = new Person[firstNames.Length];
        for (int ctr = 0; ctr < firstNames.Length; ctr++)
            p[ctr] = new Person(firstNames[ctr]);

        return p;
    }

    public Person(String firstName)
    {
        this.FirstName = firstName;
    }

    public String FirstName;
}

// The example displays the following output:
//      Unhandled Exception: System.NullReferenceException:
//          Object reference not set to an instance of an object.
//              at Example.Main()
```

To address this problem, test the method's return value to ensure that it is not `null` before calling any of its members, as the following example does.

C#

```
using System;

public class Example
{
    public static void Main()
    {
        Person[] persons = Person.AddRange( new String[] { "Abigail",
        "Abra",
                    "Abraham", "Adrian",
        "Ariella",
                    "Arnold", "Aston", "Astor" } );
        String nameToFind = "Robert";
        Person found = Array.Find(persons, p => p.FirstName ==
nameToFind);
        if (found != null)
            Console.WriteLine(found.FirstName);
        else
            Console.WriteLine("{0} not found.", nameToFind);
    }
}

public class Person
{
    public static Person[] AddRange(String[] firstNames)
    {
        Person[] p = new Person[firstNames.Length];
        for (int ctr = 0; ctr < firstNames.Length; ctr++)
            p[ctr] = new Person(firstNames[ctr]);

        return p;
    }

    public Person(String firstName)
    {
        this.FirstName = firstName;
    }

    public String FirstName;
}

// The example displays the following output:
//      Robert not found
```

- You're using an expression (for example, you're chaining a list of methods or properties together) to retrieve a value and, although you're checking whether the value is `null`, the runtime still throws a `NullReferenceException` exception. This occurs because one of the intermediate values in the expression returns `null`. As a result, your test for `null` is never evaluated.

The following example defines a `Pages` object that caches information about web pages, which are presented by `Page` objects. The `Example.Main` method checks whether the current web page has a non-null title and, if it does, displays the title. Despite this check, however, the method throws a `NullReferenceException` exception.

```
C#  
  
using System;  
  
public class Example  
{  
    public static void Main()  
    {  
        var pages = new Pages();  
        if (! String.IsNullOrEmpty(pages.CurrentPage.Title)) {  
            String title = pages.CurrentPage.Title;  
            Console.WriteLine("Current title: '{0}'", title);  
        }  
    }  
}  
  
public class Pages  
{  
    Page[] page = new Page[10];  
    int ctr = 0;  
  
    public Page CurrentPage  
    {  
        get { return page[ctr]; }  
        set {  
            // Move all the page objects down to accommodate the new one.  
            if (ctr > page.GetUpperBound(0)) {  
                for (int ndx = 1; ndx <= page.GetUpperBound(0); ndx++)  
                    page[ndx - 1] = page[ndx];  
            }  
            page[ctr] = value;  
            if (ctr < page.GetUpperBound(0))  
                ctr++;  
        }  
    }  
  
    public Page PreviousPage  
    {  
        get {  
            if (ctr == 0) {  
                if (page[0] == null)  
                    return null;  
                else  
                    return page[0];  
            }  
            else {  
                if (ctr > 0)  
                    return page[ctr - 1];  
                else  
                    return null;  
            }  
        }  
    }  
}
```

```
        ctr--;
        return page[ctr + 1];
    }
}
}

public class Page
{
    public Uri URL;
    public String Title;
}

// The example displays the following output:
//     Unhandled Exception:
//         System.NullReferenceException: Object reference not set to an
//             instance of an object.
//             at Example.Main()
```

The exception is thrown because `pages.CurrentPage` returns `null` if no page information is stored in the cache. This exception can be corrected by testing the value of the `currentPage` property before retrieving the current `Page` object's `Title` property, as the following example does:

C#

```
using System;

public class Example
{
    public static void Main()
    {
        var pages = new Pages();
        Page current = pages.CurrentPage;
        if (current != null) {
            String title = current.Title;
            Console.WriteLine("Current title: '{0}'", title);
        }
        else {
            Console.WriteLine("There is no page information in the
cache.");
        }
    }
}

// The example displays the following output:
//     There is no page information in the cache.
```

- You're enumerating the elements of an array that contains reference types, and your attempt to process one of the elements throws a `NullReferenceException` exception.

The following example defines a string array. A `for` statement enumerates the elements in the array and calls each string's `Trim` method before displaying the string.

C#

```
using System;

public class Example
{
    public static void Main()
    {
        String[] values = { "one", null, "two" };
        for (int ctr = 0; ctr <= values.GetUpperBound(0); ctr++)
            Console.WriteLine("{0}{1}", values[ctr].Trim(),
                ctr == values.GetUpperBound(0) ? "" : ", ");
        Console.WriteLine();
    }
}

// The example displays the following output:
//     Unhandled Exception:
//         System.NullReferenceException: Object reference not set to an
//             instance of an object.
//             at Example.Main()
```

This exception occurs if you assume that each element of the array must contain a non-null value, and the value of the array element is in fact `null`. The exception can be eliminated by testing whether the element is `null` before performing any operation on that element, as the following example shows.

C#

```
using System;

public class Example
{
    public static void Main()
    {
        String[] values = { "one", null, "two" };
        for (int ctr = 0; ctr <= values.GetUpperBound(0); ctr++)
            Console.WriteLine("{0}{1}",
                values[ctr] != null ? values[ctr].Trim() : "",
                ctr == values.GetUpperBound(0) ? "" : ", ");
        Console.WriteLine();
    }
}

// The example displays the following output:
//     one, , two
```

- A [NullReferenceException](#) exception is thrown by a method that is passed `null`.

Some methods validate the arguments that are passed to them. If they do and one of the arguments is `null`, the method throws an [System.ArgumentNullException](#) exception. Otherwise, it throws a [NullReferenceException](#) exception. The following example illustrates this scenario.

C#

```
using System;
using System.Collections.Generic;

public class Example
{
    public static void Main()
    {
        List<String> names = GetData();
        PopulateNames(names);
    }

    private static void PopulateNames(List<String> names)
    {
        String[] arrNames = { "Dakota", "Samuel", "Nikita",
                             "Koani", "Saya", "Yiska", "Yumaevsky" };
        foreach (var arrName in arrNames)
            names.Add(arrName);
    }

    private static List<String> GetData()
    {
        return null;
    }
}

// The example displays output like the following:
//     Unhandled Exception: System.NullReferenceException: Object reference
//     not set to an instance of an object.
//         at Example.PopulateNames(List`1 names)
//         at Example.Main()
```

To address this issue, make sure that the argument passed to the method is not `null`, or handle the thrown exception in a `try...catch...finally` block. For more information, see [Exceptions](#).

The following Microsoft intermediate language (MSIL) instructions throw [NullReferenceException](#): `callvirt`, `cpblk`, `cobj`, `initblk`, `ldelem.<type>`, `ldelema`, `ldfld`, `ldflda`, `ldind.<type>`, `ldlen`, `stelem.<type>`, `stfld`, `stind.<type>`, `throw`, and `unbox`.

[NullReferenceException](#) uses the HRESULT COR\_E\_NULLREFERENCE, which has the value 0x80004003.

For a list of initial property values for an instance of [NullReferenceException](#), see the [NullReferenceException constructors](#).

## Handling NullReferenceException in release code

It's usually better to avoid a NullReferenceException than to handle it after it occurs. Handling an exception can make your code harder to maintain and understand, and can sometimes introduce other bugs. A NullReferenceException is often a non-recoverable error. In these cases, letting the exception stop the app might be the best alternative.

However, there are many situations where handling the error can be useful:

- Your app can ignore objects that are null. For example, if your app retrieves and processes records in a database, you might be able to ignore some number of bad records that result in null objects. Recording the bad data in a log file or in the application UI might be all you have to do.
- You can recover from the exception. For example, a call to a web service that returns a reference type might return null if the connection is lost or the connection times out. You can attempt to reestablish the connection and try the call again.
- You can restore the state of your app to a valid state. For example, you might be performing a multi-step task that requires you to save information to a data store before you call a method that throws a NullReferenceException. If the uninitialized object would corrupt the data record, you can remove the previous data before you close the app.
- You want to report the exception. For example, if the error was caused by a mistake from the user of your app, you can generate a message to help them supply the correct information. You can also log information about the error to help you fix the problem. Some frameworks, like ASP.NET, have a high-level exception handler that captures all errors so that the app never crashes; in that case, logging the exception might be the only way you can know that it occurs.

## Constructors

### [NullReferenceException\(\)](#)

Initializes a new instance of the [NullReferenceException](#) class, setting the [Message](#) property of the new instance to a system-supplied message that describes the error, such as "The value

'null' was found where an instance of an object was required."  
This message takes into account the current system culture.

<a href="#">NullReferenceException(SerializationInfo, StreamingContext)</a>	Initializes a new instance of the <a href="#">NullReferenceException</a> class with serialized data.
<a href="#">NullReferenceException(String)</a>	Initializes a new instance of the <a href="#">NullReferenceException</a> class with a specified error message.
<a href="#">NullReferenceException(String, Exception)</a>	Initializes a new instance of the <a href="#">NullReferenceException</a> class with a specified error message and a reference to the inner exception that is the cause of this exception.

## Properties

<a href="#">Data</a>	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from <a href="#">Exception</a> )
<a href="#">HelpLink</a>	Gets or sets a link to the help file associated with this exception. (Inherited from <a href="#">Exception</a> )
<a href="#">HResult</a>	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from <a href="#">Exception</a> )
<a href="#">InnerException</a>	Gets the <a href="#">Exception</a> instance that caused the current exception. (Inherited from <a href="#">Exception</a> )
<a href="#">Message</a>	Gets a message that describes the current exception. (Inherited from <a href="#">Exception</a> )
<a href="#">Source</a>	Gets or sets the name of the application or the object that causes the error. (Inherited from <a href="#">Exception</a> )
<a href="#">StackTrace</a>	Gets a string representation of the immediate frames on the call stack. (Inherited from <a href="#">Exception</a> )
<a href="#">TargetSite</a>	Gets the method that throws the current exception. (Inherited from <a href="#">Exception</a> )

## Methods

<a href="#">Equals(Object)</a>	Determines whether the specified object is equal to the current object. (Inherited from <a href="#">Object</a> )
<a href="#">GetBaseException()</a>	When overridden in a derived class, returns the <a href="#">Exception</a> that is the root cause of one or more subsequent exceptions. (Inherited from <a href="#">Exception</a> )
<a href="#">GetHashCode()</a>	Serves as the default hash function. (Inherited from <a href="#">Object</a> )
<a href="#">GetObjectData(SerializationInfo, StreamingContext)</a>	When overridden in a derived class, sets the <a href="#">SerializationInfo</a> with information about the exception. (Inherited from <a href="#">Exception</a> )
<a href="#">GetType()</a>	Gets the runtime type of the current instance. (Inherited from <a href="#">Exception</a> )
<a href="#">MemberwiseClone()</a>	Creates a shallow copy of the current <a href="#">Object</a> . (Inherited from <a href="#">Object</a> )
<a href="#">ToString()</a>	Creates and returns a string representation of the current exception. (Inherited from <a href="#">Exception</a> )

## Events

<a href="#">SerializeObjectState</a>	<b>Obsolete.</b> Occurs when an exception is serialized to create an exception state object that contains serialized data about the exception. (Inherited from <a href="#">Exception</a> )
--------------------------------------	--

## Applies to

Product	Versions
<b>.NET</b>	Core 1.0, Core 1.1, Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8
<b>.NET Framework</b>	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
<b>.NET Standard</b>	1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 2.0, 2.1
<b>UWP</b>	10.0
<b>Xamarin.iOS</b>	10.8
<b>Xamarin.Mac</b>	3.0

## See also

- [Exception](#)
- [Handling and Throwing Exceptions](#)

# ArgumentNullException Class

Reference

## Definition

Namespace: [System](#)

Assembly: System.Runtime.dll

The exception that is thrown when a null reference (Nothing in Visual Basic) is passed to a method that does not accept it as a valid argument.

C#

```
public class ArgumentNullException : ArgumentException
```

Inheritance [Object](#) → [Exception](#) → [SystemException](#) → [ArgumentException](#) → [ArgumentNullException](#)

## Remarks

An [ArgumentNullException](#) exception is thrown when a method is invoked and at least one of the passed arguments is `null` but should never be `null`.

An [ArgumentNullException](#) exception is thrown at run time in the following two major circumstances, both of which reflect developer error:

- An uninstantiated object is passed to a method. To prevent the error, instantiate the object.
- An object returned from a method call is then passed as an argument to a second method, but the value of the original returned object is `null`. To prevent the error, check for a return value that is `null` and call the second method only if the return value is not `null`.

[ArgumentNullException](#) behaves identically to [ArgumentException](#). It is provided so that application code can differentiate between exceptions caused by `null` arguments and exceptions caused by arguments that are not `null`. For errors caused by arguments that are not `null`, see [ArgumentOutOfRangeException](#).

[ArgumentNullException](#) uses the HRESULT E\_POINTER, which has the value 0x80004003.

For a list of initial property values for an instance of [ArgumentNullException](#), see the [ArgumentNullException constructors](#).

## Constructors

<a href="#">ArgumentNullException()</a>	Initializes a new instance of the <a href="#">ArgumentNullException</a> class.
<a href="#">ArgumentNullException(SerializationInfo, StreamingContext)</a>	Initializes a new instance of the <a href="#">ArgumentNullException</a> class with serialized data.
<a href="#">ArgumentNullException(String)</a>	Initializes a new instance of the <a href="#">ArgumentNullException</a> class with the name of the parameter that causes this exception.
<a href="#">ArgumentNullException(String, Exception)</a>	Initializes a new instance of the <a href="#">ArgumentNullException</a> class with a specified error message and the exception that is the cause of this exception.
<a href="#">ArgumentNullException(String, String)</a>	Initializes an instance of the <a href="#">ArgumentNullException</a> class with a specified error message and the name of the parameter that causes this exception.

## Properties

<a href="#">Data</a>	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from <a href="#">Exception</a> )
<a href="#">HelpLink</a>	Gets or sets a link to the help file associated with this exception. (Inherited from <a href="#">Exception</a> )
<a href="#">HResult</a>	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from <a href="#">Exception</a> )
<a href="#">InnerException</a>	Gets the <a href="#">Exception</a> instance that caused the current exception. (Inherited from <a href="#">Exception</a> )
<a href="#">Message</a>	Gets the error message and the parameter name, or only the error message if no parameter name is set. (Inherited from <a href="#">ArgumentException</a> )
<a href="#">ParamName</a>	Gets the name of the parameter that causes this exception. (Inherited from <a href="#">ArgumentException</a> )
<a href="#">Source</a>	Gets or sets the name of the application or the object that causes the error.

(Inherited from [Exception](#))

<a href="#">StackTrace</a>	Gets a string representation of the immediate frames on the call stack. (Inherited from <a href="#">Exception</a> )
<a href="#">TargetSite</a>	Gets the method that throws the current exception. (Inherited from <a href="#">Exception</a> )

## Methods

<a href="#">Equals(Object)</a>	Determines whether the specified object is equal to the current object. (Inherited from <a href="#">Object</a> )
<a href="#">GetBaseException()</a>	When overridden in a derived class, returns the <a href="#">Exception</a> that is the root cause of one or more subsequent exceptions. (Inherited from <a href="#">Exception</a> )
<a href="#">GetHashCode()</a>	Serves as the default hash function. (Inherited from <a href="#">Object</a> )
<a href="#">GetObjectData(SerializationInfo, StreamingContext)</a>	Sets the <a href="#">SerializationInfo</a> object with the parameter name and additional exception information. (Inherited from <a href="#">ArgumentException</a> )
<a href="#">GetType()</a>	Gets the runtime type of the current instance. (Inherited from <a href="#">Exception</a> )
<a href="#">MemberwiseClone()</a>	Creates a shallow copy of the current <a href="#">Object</a> . (Inherited from <a href="#">Object</a> )
<a href="#">ThrowIfNull(Object, String)</a>	Throws an <a href="#">ArgumentNullException</a> if argument is null.
<a href="#">ThrowIfNull(Void*, String)</a>	Throws an <a href="#">ArgumentNullException</a> if argument is null.
<a href="#">ToString()</a>	Creates and returns a string representation of the current exception. (Inherited from <a href="#">Exception</a> )

## Events

<a href="#">SerializeObjectState</a>	<b>Obsolete.</b> Occurs when an exception is serialized to create an exception state object that contains serialized data about the exception. (Inherited from <a href="#">Exception</a> )
--------------------------------------	--

# Applies to

Product	Versions
.NET	Core 1.0, Core 1.1, Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 2.0, 2.1
UWP	10.0
Xamarin.iOS	10.8
Xamarin.Mac	3.0

## See also

- [Exception](#)
- [Handling and Throwing Exceptions](#)

# InvalidOperationException Class

Reference

## Definition

Namespace: [System](#)

Assembly: System.Runtime.dll

The exception that is thrown when a method call is invalid for the object's current state.

C#

```
public class InvalidOperationException : SystemException
```

Inheritance Object → [Exception](#) → [SystemException](#) → InvalidOperationException

Derived [Microsoft.Extensions.DiagnosticAdapter.Internal.InvalidProxyOperationException](#)

[System.Data.Linq.DuplicateKeyException](#)

[System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException](#)

[System.Data.Services.Client.DataServiceClientException](#)

[System.Data.Services.Client.DataServiceQueryException](#)

[More...](#)

## Remarks

[InvalidOperationException](#) is used in cases when the failure to invoke a method is caused by reasons other than invalid arguments. Typically, it is thrown when the state of an object cannot support the method call. For example, an [InvalidOperationException](#) exception is thrown by methods such as:

- [IEnumerator.MoveNext](#) if objects of a collection are modified after the enumerator is created. For more information, see [Changing a collection while iterating it](#).
- [ResourceSet.GetString](#) if the resource set is closed before the method call is made.
- [XContainer.Add](#), if the object or objects to be added would result in an incorrectly structured XML document.

- A method that attempts to manipulate the UI from a thread that is not the main or UI thread.

### ① Important

Because the **InvalidOperationException** exception can be thrown in a wide variety of circumstances, it is important to read the exception message returned by the **Message** property.

In this section:

[Some common causes of InvalidOperationException exceptions](#)

[Updating a UI thread from a non-UI thread](#)

[Changing a collection while iterating it](#)

[Sorting an array or collection whose objects cannot be compared](#)

[Casting a Nullable<T> that is null to its underlying type](#)

[Calling a System.Linq.Enumerable method on an empty collection](#)

[Calling Enumerable.Single or Enumerable.SingleOrDefault on a sequence without one element](#)

[Dynamic cross-application domain field access](#)

[Throwing an InvalidOperationException exception](#)

[Miscellaneous information](#)

## Some common causes of InvalidOperationException exceptions

The following sections show how some common cases in which an **InvalidOperationException** exception is thrown in an app. How you handle the issue depends on the specific situation. Most commonly, however, the exception results from developer error, and the **InvalidOperationException** exception can be anticipated and avoided.

### Updating a UI thread from a non-UI thread

Often, worker threads are used to perform some background work that involves gathering data to be displayed in an application's user interface. However, most GUI (graphical user interface) application frameworks for .NET, such as Windows Forms and Windows Presentation Foundation (WPF), let you access GUI objects only from the thread that creates and manages the UI (the Main or UI thread). An **InvalidOperationException** is thrown when you try to access a UI element from a thread

other than the UI thread. The text of the exception message is shown in the following table.

Application Type	Message
WPF app	The calling thread cannot access this object because a different thread owns it.
UWP app	The application called an interface that was marshaled for a different thread.
Windows Forms app	Cross-thread operation not valid: Control 'TextBox1' accessed from a thread other than the thread it was created on.

UI frameworks for .NET implement a *dispatcher* pattern that includes a method to check whether a call to a member of a UI element is being executed on the UI thread, and other methods to schedule the call on the UI thread:

- In WPF apps, call the [Dispatcher.CheckAccess](#) method to determine if a method is running on a non-UI thread. It returns `true` if the method is running on the UI thread and `false` otherwise. Call one of the overloads of the [Dispatcher.Invoke](#) method to schedule the call on the UI thread.
- In UWP apps, check the [CoreDispatcher.HasThreadAccess](#) property to determine if a method is running on a non-UI thread. Call the [CoreDispatcher.RunAsync](#) method to execute a delegate that updates the UI thread.
- In Windows Forms apps, use the [Control.InvokeRequired](#) property to determine if a method is running on a non-UI thread. Call one of the overloads of the [Control.Invoke](#) method to execute a delegate that updates the UI thread.

The following examples illustrate the [InvalidOperationException](#) exception that is thrown when you attempt to update a UI element from a thread other than the thread that created it. Each example requires that you create two controls:

- A text box control named `textBox1`. In a Windows Forms app, you should set its [Multiline](#) property to `true`.
- A button control named `threadExampleBtn`. The example provides a handler, `ThreadsExampleBtn_Click`, for the button's `Click` event.

In each case, the `threadExampleBtn_Click` event handler calls the `DoSomeWork` method twice. The first call runs synchronously and succeeds. But the second call, because it

runs asynchronously on a thread pool thread, attempts to update the UI from a non-UI thread. This results in a [InvalidOperationException](#) exception.

## WPF and UWP apps

C#

```
private async void threadExampleBtn_Click(object sender, RoutedEventArgs e)
{
    textBox1.Text = String.Empty;

    textBox1.Text = "Simulating work on UI thread.\n";
    DoSomeWork(20);
    textBox1.Text += "Work completed...\n";

    textBox1.Text += "Simulating work on non-UI thread.\n";
    await Task.Run(() => DoSomeWork(1000));
    textBox1.Text += "Work completed...\n";
}

private async void DoSomeWork(int milliseconds)
{
    // Simulate work.
    await Task.Delay(milliseconds);

    // Report completion.
    var msg = String.Format("Some work completed in {0} ms.\n", milliseconds);
    textBox1.Text += msg;
}
```

The following version of the `DoSomeWork` method eliminates the exception in a WPF app.

C#

```
private async void DoSomeWork(int milliseconds)
{
    // Simulate work.
    await Task.Delay(milliseconds);

    // Report completion.
    bool uiAccess = textBox1.Dispatcher.CheckAccess();
    String msg = String.Format("Some work completed in {0} ms. on {1}UI
thread\n",
                                milliseconds, uiAccess ? String.Empty : "non-");
    if (uiAccess)
        textBox1.Text += msg;
    else
        textBox1.Dispatcher.Invoke(() => { textBox1.Text += msg; });
}
```

The following version of the `DoSomeWork` method eliminates the exception in a UWP app.

C#

```
private async void DoSomeWork(int milliseconds)
{
    // Simulate work.
    await Task.Delay(milliseconds);

    // Report completion.
    bool uiAccess = textBox1.Dispatcher.HasThreadAccess;
    String msg = String.Format("Some work completed in {0} ms. on {1}UI
thread\n",
                                milliseconds, uiAccess ? String.Empty : "non-");
    if (uiAccess)
        textBox1.Text += msg;
    else
        await textBox1.Dispatcher.RunAsync(CoreDispatcherPriority.Normal, () => { textBox1.Text += msg; });
}
```

## Windows Forms apps

C#

```
List<String> lines = new List<String>();

private async void threadExampleBtn_Click(object sender, EventArgs e)
{
    textBox1.Text = String.Empty;
    lines.Clear();

    lines.Add("Simulating work on UI thread.");
    textBox1.Lines = lines.ToArray();
    DoSomeWork(20);

    lines.Add("Simulating work on non-UI thread.");
    textBox1.Lines = lines.ToArray();
    await Task.Run(() => DoSomeWork(1000));

    lines.Add("ThreadsExampleBtn_Click completes. ");
    textBox1.Lines = lines.ToArray();
}

private async void DoSomeWork(int milliseconds)
{
    // simulate work
    await Task.Delay(milliseconds);

    // report completion
}
```

```

        lines.Add(String.Format("Some work completed in {0} ms on UI thread.", 
milliseconds));
        textBox1.Lines = lines.ToArray();
    }
}

```

The following version of the `DoSomeWork` method eliminates the exception in a Windows Forms app.

C#

```

private async void DoSomeWork(int milliseconds)
{
    // simulate work
    await Task.Delay(milliseconds);

    // Report completion.
    bool uiMarshal = textBox1.InvokeRequired;
    String msg = String.Format("Some work completed in {0} ms. on {1}UI
thread\n",
                                milliseconds, uiMarshal ? String.Empty :
"non-");
    lines.Add(msg);

    if (uiMarshal) {
        textBox1.Invoke(new Action(() => { textBox1.Lines = lines.ToArray();
}));
    }
    else {
        textBox1.Lines = lines.ToArray();
    }
}

```

## Changing a collection while iterating it

The `foreach` statement in C#, `for...in` in F#, or `For Each` statement in Visual Basic is used to iterate the members of a collection and to read or modify its individual elements. However, it can't be used to add or remove items from the collection. Doing this throws an `InvalidOperationException` exception with a message that is similar to, "Collection was modified; enumeration operation may not execute."

The following example iterates a collection of integers attempts to add the square of each integer to the collection. The example throws an `InvalidOperationException` with the first call to the `List<T>.Add` method.

C#

```
using System;
using System.Collections.Generic;

public class Example
{
    public static void Main()
    {
        var numbers = new List<int>() { 1, 2, 3, 4, 5 };
        foreach (var number in numbers)
        {
            int square = (int) Math.Pow(number, 2);
            Console.WriteLine("{0}^{1}", number, square);
            Console.WriteLine("Adding {0} to the collection...\n", square);
            numbers.Add(square);
        }
    }
}

// The example displays the following output:
//      1^1
//      Adding 1 to the collection...
//
//
//      Unhandled Exception: System.InvalidOperationException: Collection was
modified;
//          enumeration operation may not execute.
//          at
System.ThrowHelper.ThrowInvalidOperationException(ExceptionResource re-
source)
//          at System.Collections.Generic.List`1.Enumerator.MoveNextRare()
//          at Example.Main()
```

You can eliminate the exception in one of two ways, depending on your application logic:

- If elements must be added to the collection while iterating it, you can iterate it by index using the `for (for..to in F#)` statement instead of `foreach`, `for...in`, or `For Each`. The following example uses the `for` statement to add the square of numbers in the collection to the collection.

C#

```
using System;
using System.Collections.Generic;

public class Example
{
    public static void Main()
    {
        var numbers = new List<int>() { 1, 2, 3, 4, 5 };

        int upperBound = numbers.Count - 1;
```

```

        for (int ctr = 0; ctr <= upperBound; ctr++) {
            int square = (int) Math.Pow(numbers[ctr], 2);
            Console.WriteLine("{0}^{1}", numbers[ctr], square);
            Console.WriteLine("Adding {0} to the collection...\n",
square);
            numbers.Add(square);
        }

        Console.WriteLine("Elements now in the collection: ");
        foreach (var number in numbers)
            Console.Write("{0}    ", number);
    }
}

// The example displays the following output:
//    1^1
//    Adding 1 to the collection...
//
//    2^4
//    Adding 4 to the collection...
//
//    3^9
//    Adding 9 to the collection...
//
//    4^16
//    Adding 16 to the collection...
//
//    5^25
//    Adding 25 to the collection...
//
//    Elements now in the collection:
//    1    2    3    4    5    1    4    9    16    25

```

Note that you must establish the number of iterations before iterating the collection either by using a counter inside the loop that will exit the loop appropriately, by iterating backward, from `Count - 1` to 0, or, as the example does, by assigning the number of elements in the array to a variable and using it to establish the upper bound of the loop. Otherwise, if an element is added to the collection on every iteration, an endless loop results.

- If it is not necessary to add elements to the collection while iterating it, you can store the elements to be added in a temporary collection that you add when iterating the collection has finished. The following example uses this approach to add the square of numbers in a collection to a temporary collection, and then to combine the collections into a single array object.

C#

```

using System;
using System.Collections.Generic;

```

```
public class Example
{
    public static void Main()
    {
        var numbers = new List<int>() { 1, 2, 3, 4, 5 };
        var temp = new List<int>();

        // Square each number and store it in a temporary collection.
        foreach (var number in numbers)
        {
            int square = (int) Math.Pow(number, 2);
            temp.Add(square);
        }

        // Combine the numbers into a single array.
        int[] combined = new int[numbers.Count + temp.Count];
        Array.Copy(numbers.ToArray(), 0, combined, 0, numbers.Count);
        Array.Copy(temp.ToArray(), 0, combined, numbers.Count,
temp.Count);

        // Iterate the array.
        foreach (var value in combined)
            Console.Write("{0}    ", value);
    }
}

// The example displays the following output:
//      1    2    3    4    5    1    4    9    16    25
```

## Sorting an array or collection whose objects cannot be compared

General-purpose sorting methods, such as the [Array.Sort\(Array\)](#) method or the [List<T>.Sort\(\)](#) method, usually require that at least one of the objects to be sorted implement the [IComparable<T>](#) or the [IComparable](#) interface. If not, the collection or array cannot be sorted, and the method throws an [InvalidOperationException](#) exception. The following example defines a `Person` class, stores two `Person` objects in a generic `List<T>` object, and attempts to sort them. As the output from the example shows, the call to the `List<T>.Sort()` method throws an [InvalidOperationException](#).

C#

```
using System;
using System.Collections.Generic;

public class Person
{
    public Person(String fName, String lName)
    {
        FirstName = fName;
        LastName = lName;
    }
```

```
public String FirstName { get; set; }
public String LastName { get; set; }
}

public class Example
{
    public static void Main()
    {
        var people = new List<Person>();

        people.Add(new Person("John", "Doe"));
        people.Add(new Person("Jane", "Doe"));
        people.Sort();
        foreach (var person in people)
            Console.WriteLine("{0} {1}", person.FirstName, person.LastName);
    }
}

// The example displays the following output:
//     Unhandled Exception: System.InvalidOperationException: Failed to compare two elements in the array. --->
//         System.ArgumentException: At least one object must implement IComparable.
//             at System.Collections.Comparer.Compare(Object a, Object b)
//             at System.Collections.Generic.ArraySortHelper`1.SwapIfGreater(T[]
keys, IComparer`1 comparer, Int32 a, Int32 b)
//             at
System.Collections.Generic.ArraySortHelper`1.DepthLimitedQuickSort(T[] keys,
Int32 left, Int32 right, IComparer`1 comparer, Int32 depthLimit)
//             at System.Collections.Generic.ArraySortHelper`1.Sort(T[] keys,
Int32 index, Int32 length, IComparer`1 comparer)
//             --- End of inner exception stack trace ---
//             at System.Collections.Generic.ArraySortHelper`1.Sort(T[] keys,
Int32 index, Int32 length, IComparer`1 comparer)
//             at System.Array.Sort[T](T[] array, Int32 index, Int32 length,
IComparer`1 comparer)
//             at System.Collections.Generic.List`1.Sort(Int32 index, Int32 count,
IComparer`1 comparer)
//             at Example.Main()
```

You can eliminate the exception in any of three ways:

- If you can own the type that you are trying to sort (that is, if you control its source code), you can modify it to implement the `IComparable<T>` or the `IComparable` interface. This requires that you implement either the `IComparable<T>.CompareTo` or the `CompareTo` method. Adding an interface implementation to an existing type is not a breaking change.

The following example uses this approach to provide an `IComparable<T>` implementation for the `Person` class. You can still call the collection or array's general sorting method and, as the output from the example shows, the collection sorts successfully.

C#

```
using System;
using System.Collections.Generic;

public class Person : IComparable<Person>
{
    public Person(String fName, String lName)
    {
        FirstName = fName;
        LastName = lName;
    }

    public String FirstName { get; set; }
    public String LastName { get; set; }

    public int CompareTo(Person other)
    {
        return String.Format("{0} {1}", LastName, FirstName).
            CompareTo(String.Format("{0} {1}", other.LastName,
other.FirstName));
    }
}

public class Example
{
    public static void Main()
    {
        var people = new List<Person>();

        people.Add(new Person("John", "Doe"));
        people.Add(new Person("Jane", "Doe"));
        people.Sort();
        foreach (var person in people)
            Console.WriteLine("{0} {1}", person.FirstName,
person.LastName);
    }
}
// The example displays the following output:
//      Jane Doe
//      John Doe
```

- If you cannot modify the source code for the type you are trying to sort, you can define a special-purpose sorting class that implements the `IComparer<T>` interface. You can call an overload of the `Sort` method that includes an `IComparer<T>` parameter. This approach is especially useful if you want to develop a specialized sorting class that can sort objects based on multiple criteria.

The following example uses the approach by developing a custom `PersonComparer` class that is used to sort `Person` collections. It then passes an instance of this class

to the `List<T>.Sort(IComparer<T>)` method.

C#

```
using System;
using System.Collections.Generic;

public class Person
{
    public Person(String fName, String lName)
    {
        FirstName = fName;
        LastName = lName;
    }

    public String FirstName { get; set; }
    public String LastName { get; set; }
}

public class PersonComparer : IComparer<Person>
{
    public int Compare(Person x, Person y)
    {
        return String.Format("{0} {1}", x.LastName, x.FirstName).
            CompareTo(String.Format("{0} {1}", y.LastName,
y.FirstName));
    }
}

public class Example
{
    public static void Main()
    {
        var people = new List<Person>();

        people.Add(new Person("John", "Doe"));
        people.Add(new Person("Jane", "Doe"));
        people.Sort(new PersonComparer());
        foreach (var person in people)
            Console.WriteLine("{0} {1}", person.FirstName,
person.LastName);
    }
}
// The example displays the following output:
//      Jane Doe
//      John Doe
```

- If you cannot modify the source code for the type you are trying to sort, you can create a `Comparison<T>` delegate to perform the sorting. The delegate signature is

C#

```
int Comparison<T>(T x, T y)
```

The following example uses the approach by defining a `PersonComparison` method that matches the `Comparison<T>` delegate signature. It then passes this delegate to the `List<T>.Sort(Comparison<T>)` method.

C#

```
using System;
using System.Collections.Generic;

public class Person
{
    public Person(String fName, String lName)
    {
        FirstName = fName;
        LastName = lName;
    }

    public String FirstName { get; set; }
    public String LastName { get; set; }
}

public class Example
{
    public static void Main()
    {
        var people = new List<Person>();

        people.Add(new Person("John", "Doe"));
        people.Add(new Person("Jane", "Doe"));
        people.Sort(PersonComparison);
        foreach (var person in people)
            Console.WriteLine("{0} {1}", person.FirstName,
person.LastName);
    }

    public static int PersonComparison(Person x, Person y)
    {
        return String.Format("{0} {1}", x.LastName, x.FirstName).
            CompareTo(String.Format("{0} {1}", y.LastName,
y.FirstName));
    }
}

// The example displays the following output:
//      Jane Doe
//      John Doe
```

## Casting a Nullable<T> that is null to its underlying type

Attempting to cast a `Nullable<T>` value that is `null` to its underlying type throws an `InvalidOperationException` exception and displays the error message, "**Nullable object must have a value.**"

The following example throws an `InvalidOperationException` exception when it attempts to iterate an array that includes a `Nullable(of Integer)` value.

C#

```
using System;
using System.Linq;

public class Example
{
    public static void Main()
    {
        var queryResult = new int?[] { 1, 2, null, 4 };
        var map = queryResult.Select(nullableInt => (int)nullableInt);

        // Display list.
        foreach (var num in map)
            Console.Write("{0} ", num);
        Console.WriteLine();
    }
}

// The example displays the following output:
//      1 2
//      Unhandled Exception: System.InvalidOperationException: Nullable object
//      must have a value.
//          at
System.ThrowHelper.ThrowInvalidOperationException(ExceptionResource re-
source)
//          at Example.<Main>b__0(Nullable`1 nullableInt)
//          at System.Linq.Enumerable.WhereSelectArrayIterator`2.MoveNext()
//          at Example.Main()
```

To prevent the exception:

- Use the `Nullable<T>.HasValue` property to select only those elements that are not `null`.
- Call one of the `Nullable<T>.GetValueOrDefault` overloads to provide a default value for a `null` value.

The following example does both to avoid the `InvalidOperationException` exception.

C#

```
using System;
using System.Linq;

public class Example
{
    public static void Main()
    {
        var queryResult = new int?[] { 1, 2, null, 4 };
        var numbers = queryResult.Select(nullableInt =>
(int)nullableInt.GetValueOrDefault());

        // Display list using Nullable<int>.HasValue.
        foreach (var number in numbers)
            Console.Write("{0} ", number);
        Console.WriteLine();

        numbers = queryResult.Select(nullableInt => (int)
(nullableInt.HasValue ? nullableInt : -1));
        // Display list using Nullable<int>.GetValueOrDefault.
        foreach (var number in numbers)
            Console.Write("{0} ", number);
        Console.WriteLine();
    }
}
// The example displays the following output:
//      1 2 0 4
//      1 2 -1 4
```

## Calling a System.Linq.Enumerable method on an empty collection

The `Enumerable.Average`, `Enumerable.First`, `Enumerable.Last`, `Enumerable.Max`, `Enumerable.Min`, `Enumerable.Single`, and `Enumerable.SingleOrDefault` methods perform operations on a sequence and return a single result. Some overloads of these methods throw an `InvalidOperationException` exception when the sequence is empty, while other overloads return `null`. The `Enumerable.SingleOrDefault` method also throws an `InvalidOperationException` exception when the sequence contains more than one element.

### ① Note

Most of the methods that throw an `InvalidOperationException` exception are overloads. Be sure that you understand the behavior of the overload that you choose.

The following table lists the exception messages from the `InvalidOperationException` exception objects thrown by calls to some `System.Linq.Enumerable` methods.

Method	Message
Aggregate	Sequence contains no elements
Average	
Last	
Max	
Min	
First	Sequence contains no matching element
Single	Sequence contains more than one matching element
SingleOrDefault	

How you eliminate or handle the exception depends on your application's assumptions and on the particular method you call.

- When you deliberately call one of these methods without checking for an empty sequence, you are assuming that the sequence is not empty, and that an empty sequence is an unexpected occurrence. In this case, catching or rethrowing the exception is appropriate .
- If your failure to check for an empty sequence was inadvertent, you can call one of the overloads of the [Enumerable.Any](#) overload to determine whether a sequence contains any elements.

### 💡 Tip

Calling the [Enumerable.Any<TSource>\(IQueryable<TSource>, Func<TSource, Boolean>\)](#) method before generating a sequence can improve performance if the data to be processed might contain a large number of elements or if operation that generates the sequence is expensive.

- If you've called a method such as [Enumerable.First](#), [Enumerable.Last](#), or [Enumerable.Single](#), you can substitute an alternate method, such as [Enumerable.FirstOrDefault](#), [Enumerable.LastOrDefault](#), or [Enumerable.SingleOrDefault](#), that returns a default value instead of a member of the sequence.

The examples provide additional detail.

The following example uses the [Enumerable.Average](#) method to compute the average of a sequence whose values are greater than 4. Since no values from the original array exceed 4, no values are included in the sequence, and the method throws an [InvalidOperationException](#) exception.

C#

```
using System;
using System.Linq;

public class Example
{
    public static void Main()
    {
        int[] data = { 1, 2, 3, 4 };
        var average = data.Where(num => num > 4).Average();
        Console.WriteLine("The average of numbers greater than 4 is {0}",
                          average);
    }
}
// The example displays the following output:
//     Unhandled Exception: System.InvalidOperationException: Sequence con-
// tains no elements
//         at System.Linq.Enumerable.Average(IEnumerable`1 source)
//         at Example.Main()
```

The exception can be eliminated by calling the [Any](#) method to determine whether the sequence contains any elements before calling the method that processes the sequence, as the following example shows.

C#

```
using System;
using System.Linq;

public class Example
{
    public static void Main()
    {
        int[] dbQueryResults = { 1, 2, 3, 4 };
        var moreThan4 = dbQueryResults.Where(num => num > 4);

        if(moreThan4.Any())
            Console.WriteLine("Average value of numbers greater than 4:
{0}:",
                           moreThan4.Average());
        else
            // handle empty collection
            Console.WriteLine("The dataset has no values greater than 4.");
    }
}
// The example displays the following output:
//     The dataset has no values greater than 4.
```

The `Enumerable.First` method returns the first item in a sequence or the first element in a sequence that satisfies a specified condition. If the sequence is empty and therefore does not have a first element, it throws an `InvalidOperationException` exception.

In the following example, the `Enumerable.First<TSource>(IEnumerable<TSource>, Func<TSource, Boolean>)` method throws an `InvalidOperationException` exception because the `dbQueryResults` array doesn't contain an element greater than 4.

C#

```
using System;
using System.Linq;

public class Example
{
    public static void Main()
    {
        int[] dbQueryResults = { 1, 2, 3, 4 };

        var firstNum = dbQueryResults.First(n => n > 4);

        Console.WriteLine("The first value greater than 4 is {0}",
                          firstNum);
    }
}

// The example displays the following output:
//     Unhandled Exception: System.InvalidOperationException:
//         Sequence contains no matching element
//             at System.Linq.Enumerable.First[TSource](IEnumerable`1 source,
Func`2 predicate)
//             at Example.Main()
```

You can call the `Enumerable.FirstOrDefault` method instead of `Enumerable.First` to return a specified or default value. If the method does not find a first element in the sequence, it returns the default value for that data type. The default value is `null` for a reference type, zero for a numeric data type, and `DateTime.MinValue` for the `DateTime` type.

### ① Note

Interpreting the value returned by the `Enumerable.FirstOrDefault` method is often complicated by the fact that the default value of the type can be a valid value in the sequence. In this case, you can call the `Enumerable.Any` method to determine whether the sequence has valid members before calling the `Enumerable.First` method.

The following example calls the [Enumerable.FirstOrDefault<TSource>](#) ([IEnumerable<TSource>](#), [Func<TSource,Boolean>](#)) method to prevent the [InvalidOperationException](#) exception thrown in the previous example.

C#

```
using System;
using System.Linq;

public class Example
{
    public static void Main()
    {
        int[] dbQueryResults = { 1, 2, 3, 4 };

        var firstNum = dbQueryResults.FirstOrDefault(n => n > 4);

        if (firstNum == 0)
            Console.WriteLine("No value is greater than 4.");
        else
            Console.WriteLine("The first value greater than 4 is {0}",
                firstNum);
    }
}

// The example displays the following output:
//      No value is greater than 4.
```

## Calling [Enumerable.Single](#) or [Enumerable.SingleOrDefault](#) on a sequence without one element

The [Enumerable.Single](#) method returns the only element of a sequence, or the only element of a sequence that meets a specified condition. If there are no elements in the sequence, or if there is more than one element , the method throws an [InvalidOperationException](#) exception.

You can use the [Enumerable.SingleOrDefault](#) method to return a default value instead of throwing an exception when the sequence contains no elements. However, the [Enumerable.SingleOrDefault](#) method still throws an [InvalidOperationException](#) exception when the sequence contains more than one element.

The following table lists the exception messages from the [InvalidOperationException](#) exception objects thrown by calls to the [Enumerable.Single](#) and [Enumerable.SingleOrDefault](#) methods.

Method	Message
Single	Sequence contains no matching element

Method	Message
Single	Sequence contains more than one matching element
SingleOrDefault	

In the following example, the call to the `Enumerable.Single` method throws an `InvalidOperationException` exception because the sequence doesn't have an element greater than 4.

C#

```
using System;
using System.Linq;

public class Example
{
    public static void Main()
    {
        int[] dbQueryResults = { 1, 2, 3, 4 };

        var singleObject = dbQueryResults.Single(value => value > 4);

        // Display results.
        Console.WriteLine("{0} is the only value greater than 4", singleObject);
    }
}

// The example displays the following output:
//      Unhandled Exception: System.InvalidOperationException:
//          Sequence contains no matching element
//          at System.Linq.Enumerable.Single[TSource](IEnumerable`1 source,
Func`2 predicate)
//          at Example.Main()
```

The following example attempts to prevent the `InvalidOperationException` exception thrown when a sequence is empty by instead calling the `Enumerable.SingleOrDefault` method. However, because this sequence returns multiple elements whose value is greater than 2, it also throws an `InvalidOperationException` exception.

C#

```
using System;
using System.Linq;

public class Example
{
    public static void Main()
    {
        int[] dbQueryResults = { 1, 2, 3, 4 };

        var singleObject = dbQueryResults.SingleOrDefault(value => value >
```

```
2);

    if (singleObject != 0)
        Console.WriteLine("{0} is the only value greater than 2",
                          singleObject);
    else
        // Handle an empty collection.
        Console.WriteLine("No value is greater than 2");
    }
}

// The example displays the following output:
//     Unhandled Exception: System.InvalidOperationException:
//         Sequence contains more than one matching element
//         at System.Linq.Enumerable.SingleOrDefault[TSource](IEnumerable`1
source, Func`2 predicate)
//         at Example.Main()
```

Calling the [Enumerable.Single](#) method assumes that either a sequence or the sequence that meets specified criteria contains only one element. [Enumerable.SingleOrDefault](#) assumes a sequence with zero or one result, but no more. If this assumption is a deliberate one on your part and these conditions are not met, rethrowing or catching the resulting [InvalidOperationException](#) is appropriate. Otherwise, or if you expect that invalid conditions will occur with some frequency, you should consider using some other [Enumerable](#) method, such as [FirstOrDefault](#) or [Where](#).

## Dynamic cross-application domain field access

The [OpCodes.Ldflda](#) Microsoft intermediate language (MSIL) instruction throws an [InvalidOperationException](#) exception if the object containing the field whose address you are trying to retrieve is not within the application domain in which your code is executing. The address of a field can only be accessed from the application domain in which it resides.

## Throwing an InvalidOperationException exception

You should throw an [InvalidOperationException](#) exception only when the state of your object for some reason does not support a particular method call. That is, the method call is valid in some circumstances or contexts, but is invalid in others.

If the method invocation failure is due to invalid arguments, then [ArgumentException](#) or one of its derived classes, [ArgumentNullException](#) or [ArgumentOutOfRangeException](#), should be thrown instead.

## Miscellaneous information

InvalidOperationException uses the HRESULT COR\_E\_INVALIDOPERATION, which has the value 0x80131509.

For a list of initial property values for an instance of [InvalidOperationException](#), see the [InvalidOperationException constructors](#).

## Constructors

<a href="#">InvalidOperationException()</a>	Initializes a new instance of the <a href="#">InvalidOperationException</a> class.
<a href="#">InvalidOperationException(SerializationInfo, StreamingContext)</a>	Initializes a new instance of the <a href="#">InvalidOperationException</a> class with serialized data.
<a href="#">InvalidOperationException(String)</a>	Initializes a new instance of the <a href="#">InvalidOperationException</a> class with a specified error message.
<a href="#">InvalidOperationException(String, Exception)</a>	Initializes a new instance of the <a href="#">InvalidOperationException</a> class with a specified error message and a reference to the inner exception that is the cause of this exception.

## Properties

<a href="#">Data</a>	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from <a href="#">Exception</a> )
<a href="#">HelpLink</a>	Gets or sets a link to the help file associated with this exception. (Inherited from <a href="#">Exception</a> )
<a href="#">HResult</a>	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from <a href="#">Exception</a> )
<a href="#">InnerException</a>	Gets the <a href="#">Exception</a> instance that caused the current exception. (Inherited from <a href="#">Exception</a> )
<a href="#">Message</a>	Gets a message that describes the current exception. (Inherited from <a href="#">Exception</a> )
<a href="#">Source</a>	Gets or sets the name of the application or the object that causes the error. (Inherited from <a href="#">Exception</a> )
<a href="#">StackTrace</a>	Gets a string representation of the immediate frames on the call stack. (Inherited from <a href="#">Exception</a> )

**TargetSite**

Gets the method that throws the current exception.  
(Inherited from [Exception](#))

## Methods

<a href="#">Equals(Object)</a>	Determines whether the specified object is equal to the current object. (Inherited from <a href="#">Object</a> )
<a href="#">GetBaseException()</a>	When overridden in a derived class, returns the <a href="#">Exception</a> that is the root cause of one or more subsequent exceptions. (Inherited from <a href="#">Exception</a> )
<a href="#">GetHashCode()</a>	Serves as the default hash function. (Inherited from <a href="#">Object</a> )
<a href="#">GetObjectData(SerializationInfo, StreamingContext)</a>	When overridden in a derived class, sets the <a href="#">SerializationInfo</a> with information about the exception. (Inherited from <a href="#">Exception</a> )
<a href="#">GetType()</a>	Gets the runtime type of the current instance. (Inherited from <a href="#">Exception</a> )
<a href="#">MemberwiseClone()</a>	Creates a shallow copy of the current <a href="#">Object</a> . (Inherited from <a href="#">Object</a> )
<a href="#">ToString()</a>	Creates and returns a string representation of the current exception. (Inherited from <a href="#">Exception</a> )

## Events

[SerializeObjectState](#)**Obsolete.**

Occurs when an exception is serialized to create an exception state object that contains serialized data about the exception.  
(Inherited from [Exception](#))

## Applies to

Product	Versions
<a href="#">.NET</a>	Core 1.0, Core 1.1, Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8
<a href="#">.NET Framework</a>	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1

Product	Versions
.NET Standard	1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 2.0, 2.1
UWP	10.0
Xamarin.iOS	10.8
Xamarin.Mac	3.0

## See also

- [Exception](#)
- [IEnumerator](#)
- [ResourceSet](#)
- [Handling and Throwing Exceptions](#)



# Orchestrator Standalone User Guide

DELIVERY:

AUTOMATION CLOUD

AUTOMATION SUITE

STANDALONE

RELEASE:

2023.4

TABLE OF CONTENTS

[Business Exception Vs Application Exception](#)

## Business Exception Vs Application Exception

It is important to choose the correct type of exception with which a transaction is failed, because this choice influences whether **Orchestrator** chooses to retry the transaction of the queue item or not, as follows:

- An **Application Exception** describes an error rooted in a technical issue, such as an application that is not responding.

Such a situation is, for example, a project which extracts phone numbers from an employee database, creating queue items for each of them. These items are then to be processed and inserted into a financial application. If, when the transaction is attempted, the financial application freezes, the Robot cannot find the field where it should insert the phone number, and eventually throws an error.

These kinds of issues have a chance of being solved simply by retrying the transaction, as the application can unfreeze.

- A **Business Exception** describes an error rooted in the fact that certain data which the automation project depends on is incomplete or missing.

Such a situation is, for example, a project which extracts phone numbers from an employee database, creating queue items for each of them. These items are then to be processed and inserted into a financial application. If a certain phone number is missing a digit due to human error, the queue item containing it becomes invalid. This causes the automation to throw an exception, as the Phone Number field in the financial application does not accept a queue item that contains an incomplete number.

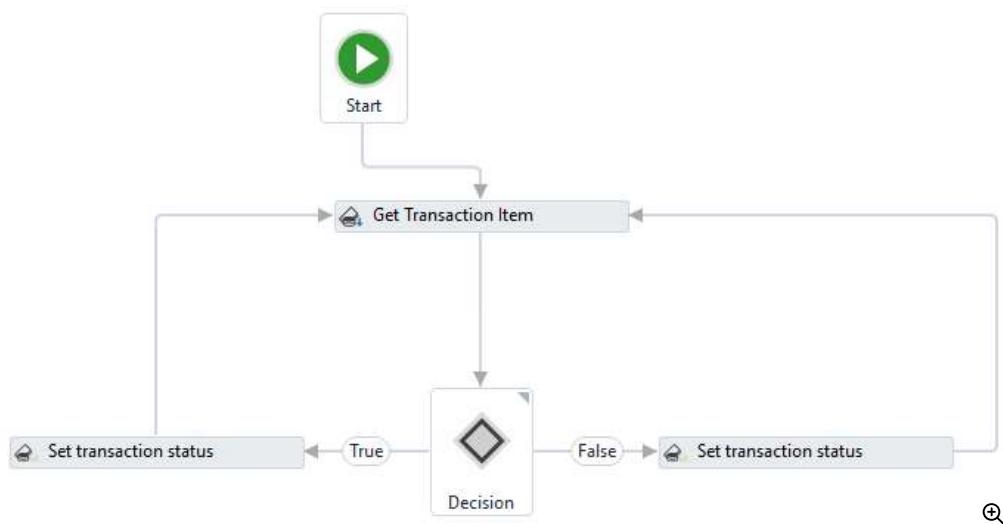
Retrying the transaction does not yield any chance of solving the issue, and there are other better courses of action, such as notifying the human user of this error.

The **Set Transaction Status** activity can be used to shape the logic of your project in a way that encapsulates this distinction in several ways:

- If the **Set Transaction Status** activity fails the transaction with an **Application Exception** and the **Auto Retry** option in the **Create Queue** page is set to **Yes** when the queue is created, the queue item is retried.
- By default, Orchestrator does **not** retry transactions which are failed due to **Business Exceptions**. This happens because an inconsistency between the transaction value and the business requirement means that there might be errors in the initial data which the queue items were created from. Additional actions might be required to fix this type of issue, and logging this type of exception can be useful.
- An **If** or **Flow Decision** activity can be used to take different courses of action if a transaction is failed with a certain type of exception, such as using the **Log Message** activity to log a custom message or the **Message Box** activity to display a window containing information

about the event.

Below you can see an example of such a project:



The screenshot below shows the mapping of the properties in the **Set Transaction Status** activity (on the left) and their corresponding fields in the **Transaction Details** window in Orchestrator.

Properties		Item Details for 1d1a17be-1526-4522-aa20-f77b2f2495b8																																		
UiPath.Core.Activities.SetTransactionStatus																																				
Common																																				
DisplayName	Set transaction status																																			
TimeoutMS	Specifies the amount of time (																																			
Input																																				
Output	(Collection)																																			
Status	<b>1</b> Failed																																			
TransactionItem	transItem																																			
Misc																																				
Private	<input type="checkbox"/>																																			
Transaction Error																																				
AssociatedFilePath	The full path of a file that is a...																																			
Details	Details regarding the failed Ti...																																			
ErrorType	<b>2</b> Business																																			
Reason	<b>3</b> "doesn't contain the letter F"																																			
<b>Item Details for 1d1a17be-1526-4522-aa20-f77b2f2495b8</b>																																				
<b>Exception</b> <ul style="list-style-type: none"> <li>Reason: doesn't contain the letter F <b>3</b></li> </ul>																																				
<table border="1"> <thead> <tr> <th>STATUS</th> <th>REVISION</th> <th>DEADLINE</th> <th>RETRYNO.</th> <th>POSTPONE</th> <th>STARTED</th> <th>ENDED</th> <th>ROBOT</th> <th>EX</th> </tr> </thead> <tbody> <tr> <td>Failed</td> <td>None</td> <td></td> <td>0</td> <td></td> <td>8 days ago</td> <td>8 days ago</td> <td>Documentat...</td> <td>Bu</td> </tr> <tr> <td><b>1</b></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>										STATUS	REVISION	DEADLINE	RETRYNO.	POSTPONE	STARTED	ENDED	ROBOT	EX	Failed	None		0		8 days ago	8 days ago	Documentat...	Bu	<b>1</b>								
STATUS	REVISION	DEADLINE	RETRYNO.	POSTPONE	STARTED	ENDED	ROBOT	EX																												
Failed	None		0		8 days ago	8 days ago	Documentat...	Bu																												
<b>1</b>																																				

The **True** branch of the **Flow Decision** activity sets the transaction status to Failed with a **Business Exception**, while the **False** branch sets it to Failed with an **Application Exception**.



Default Theme

English

# Build a Model with Business Rule Validations

Article • 06/24/2023

by Microsoft

[Download PDF](#)

This is step 3 of a free "[NerdDinner](#)" application tutorial that walks-through how to build a small, but complete, web application using ASP.NET MVC 1.

Step 3 shows how to create a model that we can use to both query and update the database for our NerdDinner application.

If you are using ASP.NET MVC 3, we recommend you follow the [Getting Started With MVC 3](#) or [MVC Music Store](#) tutorials.

## NerdDinner Step 3: Building the Model

In a model-view-controller framework the term "model" refers to the objects that represent the data of the application, as well as the corresponding domain logic that integrates validation and business rules with it. The model is in many ways the "heart" of an MVC-based application, and as we'll see later fundamentally drives the behavior of it.

The ASP.NET MVC framework supports using any data access technology, and developers can choose from a variety of rich .NET data options to implement their models including: LINQ to Entities, LINQ to SQL, NHibernate, LLBLGen Pro, SubSonic, WilsonORM, or just raw ADO.NET DataReaders or DataSets.

For our NerdDinner application we are going to use LINQ to SQL to create a simple model that corresponds fairly closely to our database design, and adds some custom validation logic and business rules. We will then implement a repository class that helps abstract away the data persistence implementation from the rest of the application, and enables us to easily unit test it.

### LINQ to SQL

LINQ to SQL is an ORM (object relational mapper) that ships as part of .NET 3.5.

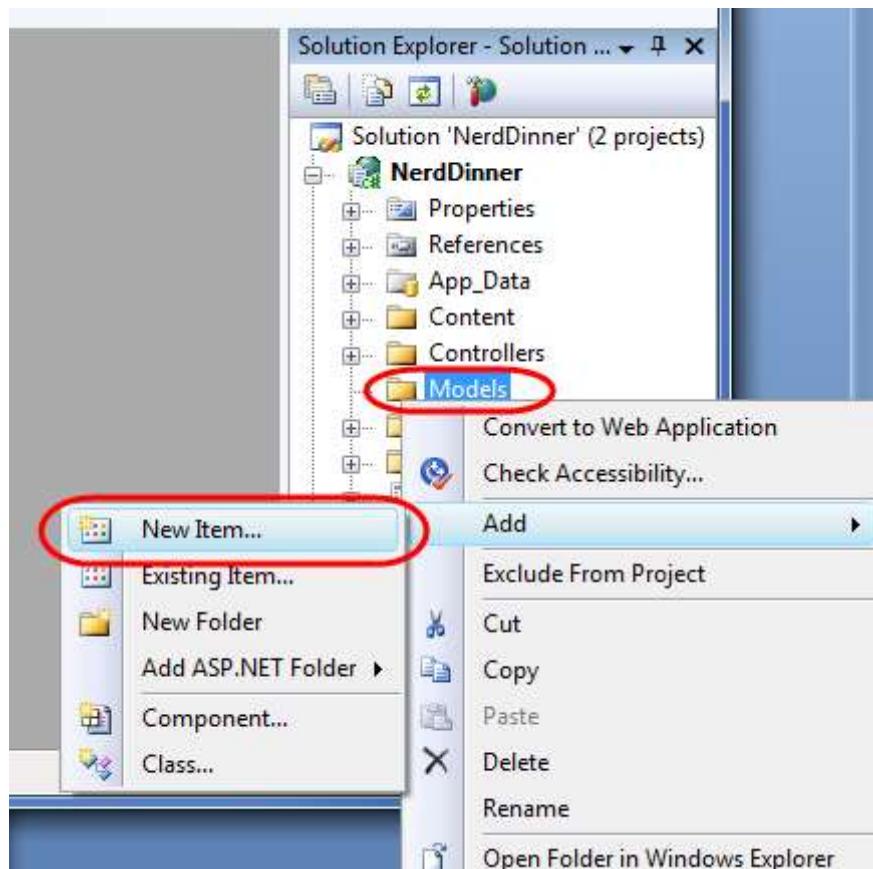
LINQ to SQL provides an easy way to map database tables to .NET classes we can code against. For our NerdDinner application we'll use it to map the Dinners and RSVP tables within our database to Dinner and RSVP classes. The columns of the Dinners and RSVP tables will correspond to properties on the Dinner and RSVP classes. Each Dinner and RSVP object will represent a separate row within the Dinners or RSVP tables in the database.

LINQ to SQL allows us to avoid having to manually construct SQL statements to retrieve and update Dinner and RSVP objects with database data. Instead, we'll define the Dinner and RSVP classes, how they map to/from the database, and the relationships between them. LINQ to SQL will then take care of generating the appropriate SQL execution logic to use at runtime when we interact and use them.

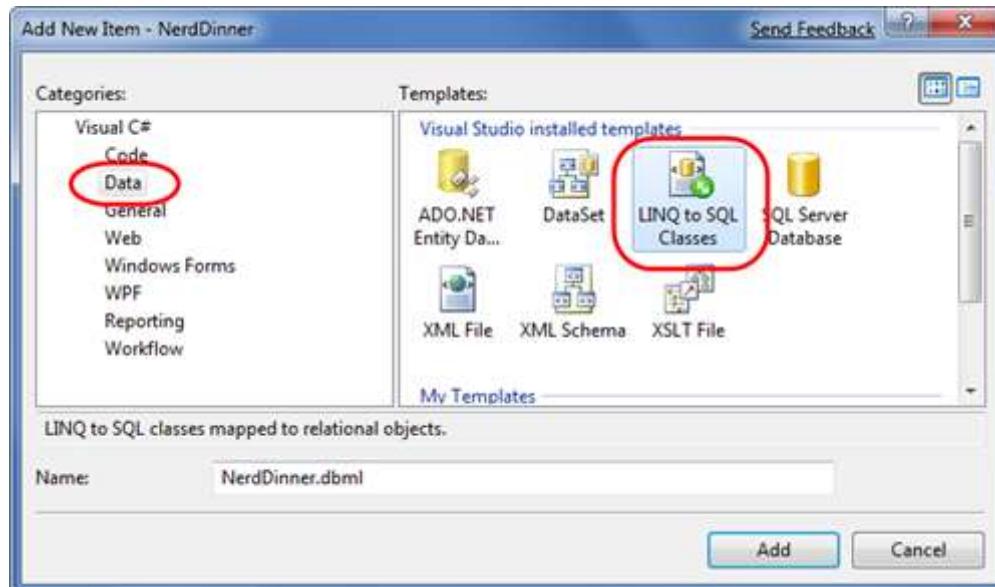
We can use the LINQ language support within VB and C# to write expressive queries that retrieve Dinner and RSVP objects from the database. This minimizes the amount of data code we need to write, and allows us to build really clean applications.

## Adding LINQ to SQL Classes to our project

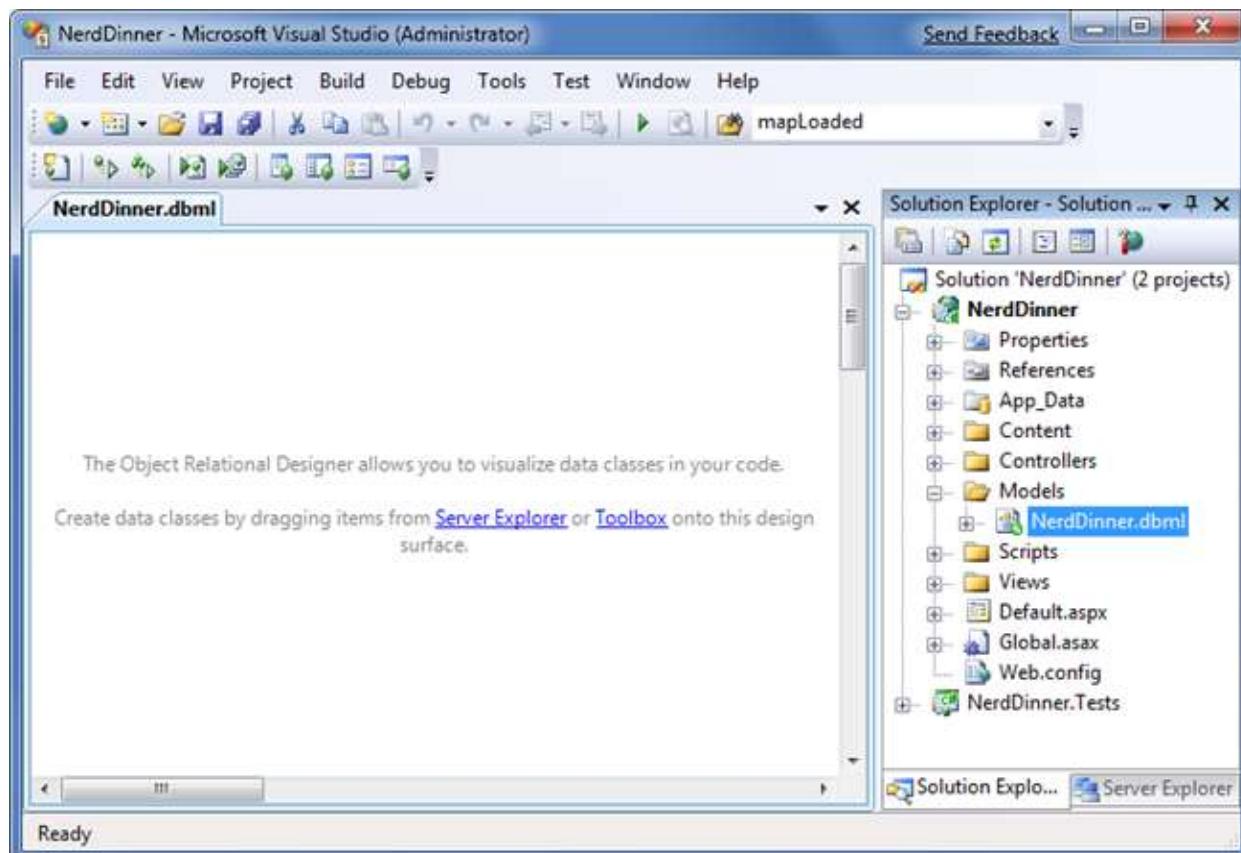
We'll begin by right-clicking on the "Models" folder within our project, and select the **Add->New Item** menu command:



This will bring up the "Add New Item" dialog. We'll filter by the "Data" category and select the "LINQ to SQL Classes" template within it:

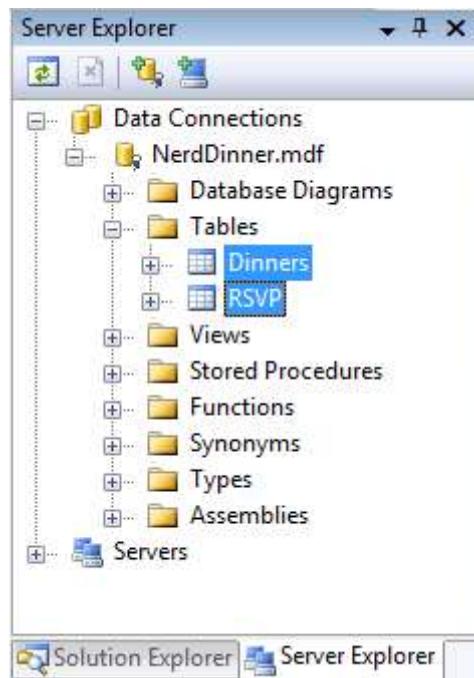


We'll name the item "NerdDinner" and click the "Add" button. Visual Studio will add a NerdDinner.dbml file under our \Models directory, and then open the LINQ to SQL object relational designer:

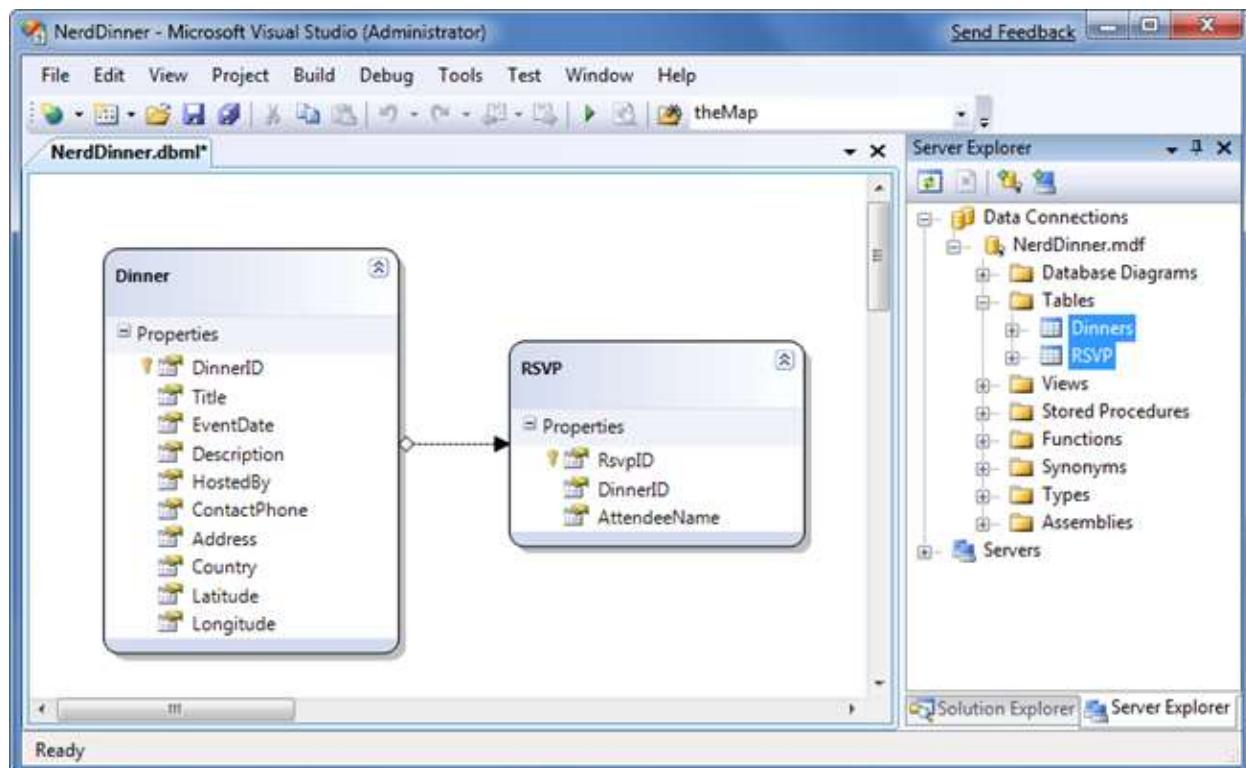


## Creating Data Model Classes with LINQ to SQL

LINQ to SQL enables us to quickly create data model classes from existing database schema. To do this we'll open the NerdDinner database in the Server Explorer, and select the Tables we want to model in it:



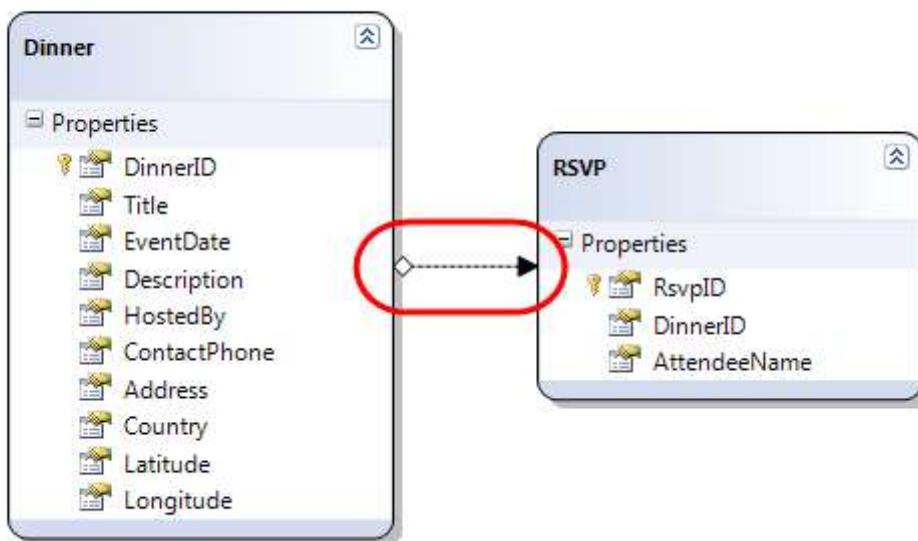
We can then drag the tables onto the LINQ to SQL designer surface. When we do this LINQ to SQL will automatically create Dinner and RSVP classes using the schema of the tables (with class properties that map to the database table columns):



By default the LINQ to SQL designer automatically "pluralizes" table and column names when it creates classes based on a database schema. For example: the "Dinners" table in our example above resulted in a "Dinner" class. This class naming helps make our models consistent with .NET naming conventions, and I usually find that having the designer fix this up convenient (especially when adding lots of tables). If you don't like the name of a class or property that the designer generates, though, you can always override it and change it to any name you want. You can do this either by editing the

entity/property name in-line within the designer or by modifying it via the property grid.

By default the LINQ to SQL designer also inspects the primary key/foreign key relationships of the tables, and based on them automatically creates default "relationship associations" between the different model classes it creates. For example, when we dragged the Dinners and RSVP tables onto the LINQ to SQL designer a one-to-many relationship association between the two was inferred based on the fact that the RSVP table had a foreign-key to the Dinners table (this is indicated by the arrow in the designer):



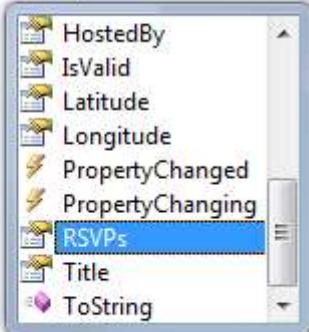
The above association will cause LINQ to SQL to add a strongly typed "Dinner" property to the RSVP class that developers can use to access the Dinner associated with a given RSVP. It will also cause the Dinner class to have a "RSVPs" collection property that enables developers to retrieve and update RSVP objects associated with a particular Dinner.

Below you can see an example of intellisense within Visual Studio when we create a new RSVP object and add it to a Dinner's RSVPs collection. Notice how LINQ to SQL automatically added a "RSVPs" collection on the Dinner object:

```
Dinner dinner = db.Dinners.Single(d => d.DinnerID == 1);

RSVP myRSVP = new RSVP();
myRSVP.AttendeeName = "ScottGu";

dinner.R
```



By adding the RSVP object to the Dinner's RSVPs collection we are telling LINQ to SQL to associate a foreign-key relationship between the Dinner and the RSVP row in our database:

```
Dinner dinner = db.Dinners.Single(d => d.DinnerID == 1);

RSVP myRSVP = new RSVP();
myRSVP.AttendeeName = "ScottGu";

dinner.RSVPs.Add(myRSVP);
```

If you don't like how the designer has modeled or named a table association, you can override it. Just click on the association arrow within the designer and access its properties via the property grid to rename, delete or modify it. For our NerdDinner application, though, the default association rules work well for the data model classes we are building and we can just use the default behavior.

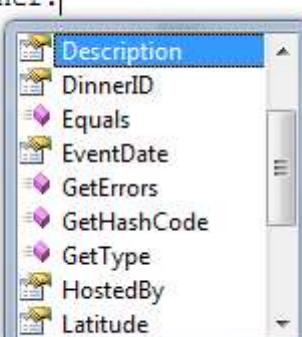
## NerdDinnerDataContext Class

Visual Studio will automatically create .NET classes that represent the models and database relationships defined using the LINQ to SQL designer. A LINQ to SQL DataContext class is also generated for each LINQ to SQL designer file added to the solution. Because we named our LINQ to SQL class item "NerdDinner", the DataContext class created will be called "NerdDinnerDataContext". This NerdDinnerDataContext class is the primary way we will interact with the database.

Our NerdDinnerDataContext class exposes two properties - "Dinners" and "RSVPs" - that represent the two tables we modeled within the database. We can use C# to write

LINQ queries against those properties to query and retrieve Dinner and RSVP objects from the database.

The following code demonstrates how to instantiate a NerdDinnerDataContext object and perform a LINQ query against it to obtain a sequence of Dinners that occur in the future. Visual Studio provides full intellisense when writing the LINQ query, and the objects returned from it are strongly-typed and also support intellisense:



```
NerdDinnerDataContext db = new NerdDinnerDataContext();

var upcomingDinners = from dinner in db.Dinners
                      where dinner.EventDate > DateTime.Now
                      orderby dinner.EventDate
                      select dinner;

foreach (Dinner dinner in upcomingDinners) {
    dinner.|
```

A screenshot of Visual Studio's Intellisense feature. A tooltip window is open over the word 'dinner' in the code. The tooltip shows a list of properties and methods for the 'Dinner' type. The 'Description' property is highlighted at the top of the list. Other visible items include DinnerID, Equals, EventDate, GetErrors, GetHashCode, GetType, HostedBy, and Latitude.

In addition to allowing us to query for Dinner and RSVP objects, a NerdDinnerDataContext also automatically tracks any changes we subsequently make to the Dinner and RSVP objects we retrieve through it. We can use this functionality to easily save the changes back to the database - without having to write any explicit SQL update code.

For example, the code below demonstrates how to use a LINQ query to retrieve a single Dinner object from the database, update two of the Dinner properties, and then save the changes back to the database:

C#

```
NerdDinnerDataContext db = new NerdDinnerDataContext();

// Retrieve Dinner object that represents row with DinnerID of 1
Dinner dinner = db.Dinners.Single(d => d.DinnerID == 1);

// Update two properties on Dinner
dinner.Title = "Changed Title";
dinner.Description = "This dinner will be fun";
```

```
// Persist changes to database  
db.SubmitChanges();
```

The NerdDinnerDataContext object in the code above automatically tracked the property changes made to the Dinner object we retrieved from it. When we called the "SubmitChanges()" method, it will execute an appropriate SQL "UPDATE" statement to the database to persist the updated values back.

## Creating a DinnerRepository Class

For small applications it is sometimes fine to have Controllers work directly against a LINQ to SQL DataContext class, and embed LINQ queries within the Controllers. As applications get larger, though, this approach becomes cumbersome to maintain and test. It can also lead to us duplicating the same LINQ queries in multiple places.

One approach that can make applications easier to maintain and test is to use a "repository" pattern. A repository class helps encapsulate data querying and persistence logic, and abstracts away the implementation details of the data persistence from the application. In addition to making application code cleaner, using a repository pattern can make it easier to change data storage implementations in the future, and it can help facilitate unit testing an application without requiring a real database.

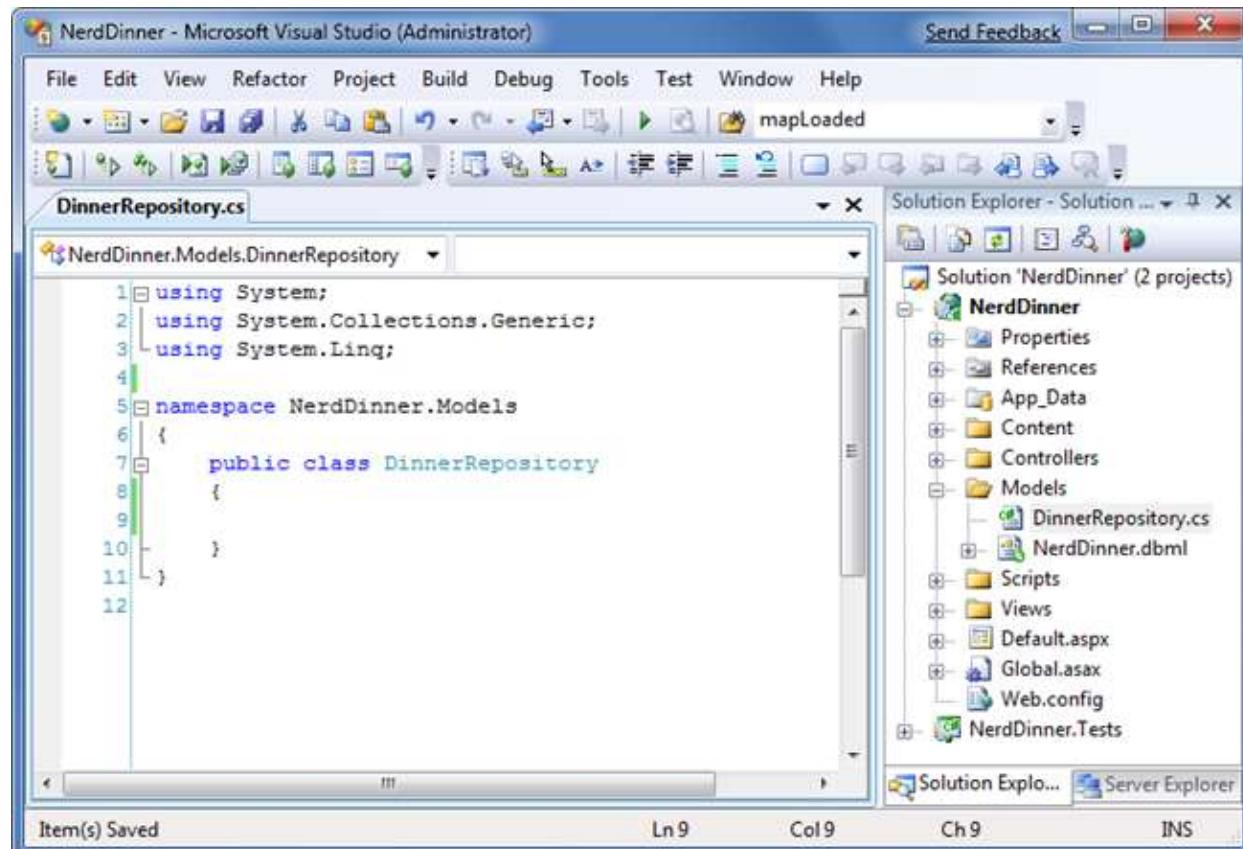
For our NerdDinner application we'll define a DinnerRepository class with the below signature:

C#

```
public class DinnerRepository {  
  
    // Query Methods  
    public IQueryable<Dinner> FindAllDinners();  
    public IQueryable<Dinner> FindUpcomingDinners();  
    public Dinner GetDinner(int id);  
  
    // Insert/Delete  
    public void Add(Dinner dinner);  
    public void Delete(Dinner dinner);  
  
    // Persistence  
    public void Save();  
}
```

*Note: Later in this chapter we'll extract an IDinnerRepository interface from this class and enable dependency injection with it on our Controllers. To begin with, though, we are going to start simple and just work directly with the DinnerRepository class.*

To implement this class we'll right-click on our "Models" folder and choose the **Add->New Item** menu command. Within the "Add New Item" dialog we'll select the "Class" template and name the file "DinnerRepository.cs":



We can then implement our DinnerRepository class using the code below:

C#

```

public class DinnerRepository {

    private NerdDinnerDataContext db = new NerdDinnerDataContext();

    //
    // Query Methods

    public IQueryable<Dinner> FindAllDinners() {
        return db.Dinners;
    }

    public IQueryable<Dinner> FindUpcomingDinners() {
        return from dinner in db.Dinners
               where dinner.EventDate > DateTime.Now
               orderby dinner.EventDate
               select dinner;
    }

    public Dinner GetDinner(int id) {
        return db.Dinners.SingleOrDefault(d => d.DinnerID == id);
    }
}

```

```
//  
// Insert/Delete Methods  
  
public void Add(Dinner dinner) {  
    db.Dinners.InsertOnSubmit(dinner);  
}  
  
public void Delete(Dinner dinner) {  
    db.RSVPs.DeleteAllOnSubmit(dinner.RSVPs);  
    db.Dinners.DeleteOnSubmit(dinner);  
}  
  
//  
// Persistence  
  
public void Save() {  
    db.SubmitChanges();  
}  
}
```

## Retrieving, Updating, Inserting and Deleting using the DinnerRepository class

Now that we've created our DinnerRepository class, let's look at a few code examples that demonstrate common tasks we can do with it:

### Querying Examples

The code below retrieves a single Dinner using the DinnerID value:

```
C#
```

```
DinnerRepository dinnerRepository = new DinnerRepository();  
  
// Retrieve specific dinner by its DinnerID  
Dinner dinner = dinnerRepository.GetDinner(5);
```

The code below retrieves all upcoming dinners and loops over them:

```
C#
```

```
DinnerRepository dinnerRepository = new DinnerRepository();  
  
// Retrieve all upcoming Dinners  
var upcomingDinners = dinnerRepository.FindUpcomingDinners();  
  
// Loop over each upcoming Dinner and print out its Title
```

```
foreach (Dinner dinner in upcomingDinners) {  
    Response.Write("Title" + dinner.Title);  
}
```

## Insert and Update Examples

The code below demonstrates adding two new dinners. Additions/modifications to the repository aren't committed to the database until the "Save()" method is called on it. LINQ to SQL automatically wraps all changes in a database transaction – so either all changes happen or none of them do when our repository saves:

C#

```
DinnerRepository dinnerRepository = new DinnerRepository();  
  
// Create First Dinner  
Dinner newDinner1 = new Dinner();  
newDinner1.Title = "Dinner with Scott";  
newDinner1.HostedBy = "ScotGu";  
newDinner1.ContactPhone = "425-703-8072";  
  
// Create Second Dinner  
Dinner newDinner2 = new Dinner();  
newDinner2.Title = "Dinner with Bill";  
newDinner2.HostedBy = "BillG";  
newDinner2.ContactPhone = "425-555-5151";  
  
// Add Dinners to Repository  
dinnerRepository.Add(newDinner1);  
dinnerRepository.Add(newDinner2);  
  
// Persist Changes  
dinnerRepository.Save();
```

The code below retrieves an existing Dinner object, and modifies two properties on it. The changes are committed back to the database when the "Save()" method is called on our repository:

C#

```
DinnerRepository dinnerRepository = new DinnerRepository();  
  
// Retrieve specific dinner by its DinnerID  
Dinner dinner = dinnerRepository.GetDinner(5);  
  
// Update Dinner properties  
dinner.Title = "Update Title";  
dinner.HostedBy = "New Owner";
```

```
// Persist changes  
dinnerRepository.Save();
```

The code below retrieves a dinner and then adds an RSVP to it. It does this using the RSVPs collection on the Dinner object that LINQ to SQL created for us (because there is a primary-key/foreign-key relationship between the two in the database). This change is persisted back to the database as a new RSVP table row when the "Save()" method is called on the repository:

C#

```
DinnerRepository dinnerRepository = new DinnerRepository();  
  
// Retrieve specific dinner by its DinnerID  
Dinner dinner = dinnerRepository.GetDinner(5);  
  
// Create a new RSVP object  
RSVP myRSVP = new RSVP();  
myRSVP.AttendeeName = "ScottGu";  
  
// Add RSVP to Dinner's RSVP Collection  
dinner.RSVPs.Add(myRSVP);  
  
// Persist changes  
dinnerRepository.Save();
```

## Delete Example

The code below retrieves an existing Dinner object, and then marks it to be deleted. When the "Save()" method is called on the repository it will commit the delete back to the database:

C#

```
DinnerRepository dinnerRepository = new DinnerRepository();  
  
// Retrieve specific dinner by its DinnerID  
Dinner dinner = dinnerRepository.GetDinner(5);  
  
// Mark dinner to be deleted  
dinnerRepository.Delete(dinner);  
  
// Persist changes  
dinnerRepository.Save();
```

# Integrating Validation and Business Rule Logic with Model Classes

Integrating validation and business rule logic is a key part of any application that works with data.

## Schema Validation

When model classes are defined using the LINQ to SQL designer, the datatypes of the properties in the data model classes correspond to the datatypes of the database table. For example: if the "EventDate" column in the Dinners table is a "datetime", the data model class created by LINQ to SQL will be of type "DateTime" (which is a built-in .NET datatype). This means you will get compile errors if you attempt to assign an integer or boolean to it from code, and it will raise an error automatically if you attempt to implicitly convert a non-valid string type to it at runtime.

LINQ to SQL will also automatically handle escaping SQL values for you when using strings - which helps protect you against SQL injection attacks when using it.

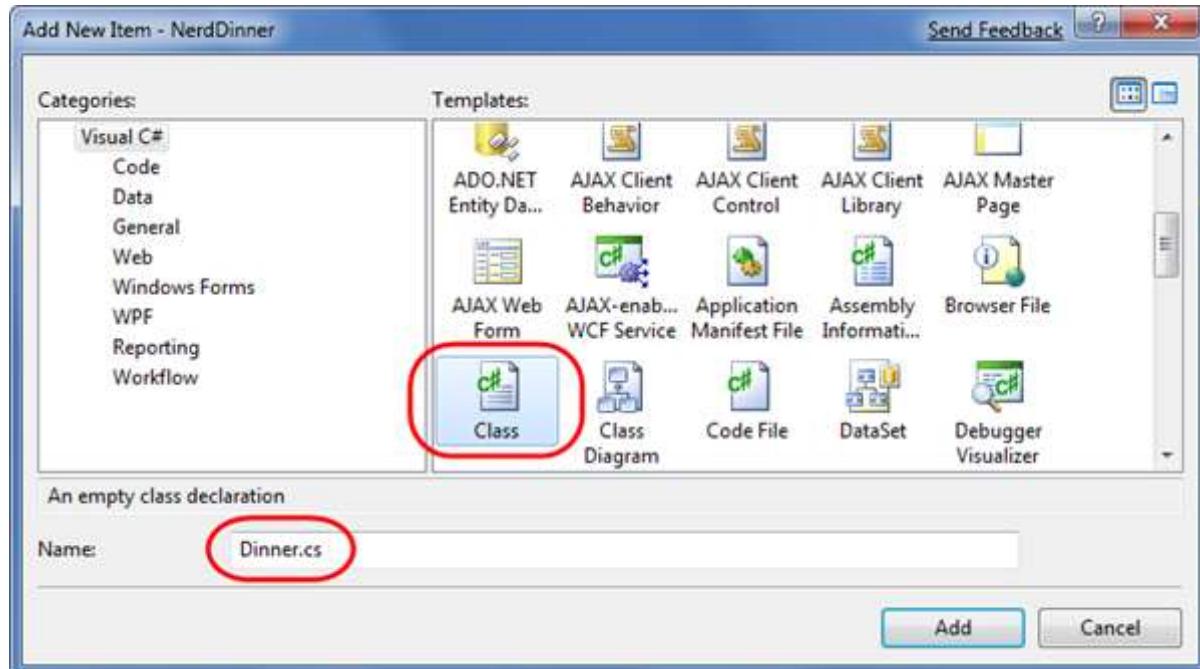
## Validation and Business Rule Logic

Schema validation is useful as a first step, but is rarely sufficient. Most real-world scenarios require the ability to specify richer validation logic that can span multiple properties, execute code, and often have awareness of a model's state (for example: is it being created /updated/deleted, or within a domain-specific state like "archived"). There are a variety of different patterns and frameworks that can be used to define and apply validation rules to model classes, and there are several .NET based frameworks out there that can be used to help with this. You can use pretty much any of them within ASP.NET MVC applications.

For the purposes of our NerdDinner application, we'll use a relatively simple and straight-forward pattern where we expose an `IsValid` property and a `GetRuleViolations()` method on our `Dinner` model object. The `IsValid` property will return true or false depending on whether the validation and business rules are all valid. The `GetRuleViolations()` method will return a list of any rule errors.

We'll implement `IsValid` and `GetRuleViolations()` for our `Dinner` model by adding a "partial class" to our project. Partial classes can be used to add methods/properties/events to classes maintained by a VS designer (like the `Dinner` class generated by the LINQ to SQL designer) and help avoid the tool from messing with our code. We can add a new partial class to our project by right-clicking on the `\Models`

folder, and then select the "Add New Item" menu command. We can then choose the "Class" template within the "Add New Item" dialog and name it Dinner.cs.



Clicking the "Add" button will add a Dinner.cs file to our project and open it within the IDE. We can then implement a basic rule/validation enforcement framework using the below code:

```
C#  
  
public partial class Dinner {  
  
    public bool IsValid {  
        get { return (GetRuleViolations().Count() == 0); }  
    }  
  
    public IEnumerable<RuleViolation> GetRuleViolations() {  
        yield break;  
    }  
  
    partial void OnValidate(ChangeAction action) {  
        if (!IsValid)  
            throw new ApplicationException("Rule violations prevent sav-  
ing");  
    }  
}  
  
public class RuleViolation {  
  
    public string ErrorMessage { get; private set; }  
    public string PropertyName { get; private set; }  
  
    public RuleViolation(string errorMessage, string propertyName) {  
        ErrorMessage = errorMessage;  
        PropertyName = propertyName;  
    }  
}
```

```
    }  
}
```

A few notes about the above code:

- The Dinner class is prefaced with a "partial" keyword – which means the code contained within it will be combined with the class generated/maintained by the LINQ to SQL designer and compiled into a single class.
- The RuleViolation class is a helper class we'll add to the project that allows us to provide more details about a rule violation.
- The Dinner.GetRuleViolations() method causes our validation and business rules to be evaluated (we'll implement them shortly). It then returns back a sequence of RuleViolation objects that provide more details about any rule errors.
- The Dinner.IsValid property provides a convenient helper property that indicates whether the Dinner object has any active RuleViolations. It can be proactively checked by a developer using the Dinner object at anytime (and does not raise an exception).
- The Dinner.OnValidate() partial method is a hook that LINQ to SQL provides that allows us to be notified anytime the Dinner object is about to be persisted within the database. Our OnValidate() implementation above ensures that the Dinner has no RuleViolations before it is saved. If it is in an invalid state it raises an exception, which will cause LINQ to SQL to abort the transaction.

This approach provides a simple framework that we can integrate validation and business rules into. For now let's add the below rules to our GetRuleViolations() method:

C#

```
public IEnumerable<RuleViolation> GetRuleViolations() {  
  
    if (String.IsNullOrEmpty>Title))  
        yield return new RuleViolation("Title required", "Title");  
  
    if (String.IsNullOrEmpty>Description))  
        yield return new RuleViolation("Description  
required", "Description");  
  
    if (String.IsNullOrEmpty(HostedBy))  
        yield return new RuleViolation("HostedBy required", "HostedBy");  
  
    if (String.IsNullOrEmpty(Address))  
        yield return new RuleViolation("Address required", "Address");  
  
    if (String.IsNullOrEmpty(Country))  
        yield return new RuleViolation("Country required", "Country");
```

```
if (String.IsNullOrEmpty(ContactPhone))
    yield return new RuleViolation("Phone# required", "ContactPhone");

if (!PhoneValidator.IsValidNumber(ContactPhone, Country))
    yield return new RuleViolation("Phone# does not match country",
"ContactPhone");

yield break;
}
```

We are using the "yield return" feature of C# to return a sequence of any RuleViolations. The first six rule checks above simply enforce that string properties on our Dinner cannot be null or empty. The last rule is a little more interesting, and calls a PhoneValidator.IsValidNumber() helper method that we can add to our project to verify that the ContactPhone number format matches the Dinner's country/region.

We can use .NET's regular expression support to implement this phone validation support. Below is a simple PhoneValidator implementation that we can add to our project that enables us to add country/region-specific Regex pattern checks:

C#

```
public class PhoneValidator {

    static IDictionary<string, Regex> countryRegex = new Dictionary<string,
Regex>() {
        { "USA", new Regex("^[2-9]\\d{2}-\\d{3}-\\d{4}$") },
        { "UK", new Regex("(^1300\\d{6}$)|(^1800|1900|1902\\d{6}$)|(^0[2|3|7|8]{1}[0-9]{8}$)|(^13\\d{4}$)|(^04\\d{2,3}\\d{6}$)") },
        { "Netherlands", new Regex("(^\\+[0-9]{2}|^\\+[0-9]{2}\\\\(0\\\\)|^\\\\([0-9]{2}\\\\)\\\\(0\\\\)|^0[0-9]{2}|^0)([0-9]{9}$|[0-9\\\\-\\\\s]{10}$)") },
    };

    public static bool IsValidNumber(string phoneNumber, string country) {

        if (country != null && countryRegex.ContainsKey(country))
            return countryRegex[country].IsMatch(phoneNumber);
        else
            return false;
    }

    public static IEnumerable<string> Countries {
        get {
            return countryRegex.Keys;
        }
    }
}
```

## Handling Validation and Business Logic Violations

Now that we've added the above validation and business rule code, any time we try to create or update a Dinner, our validation logic rules will be evaluated and enforced.

Developers can write code like below to proactively determine if a Dinner object is valid, and retrieve a list of all violations in it without raising any exceptions:

C#

```
Dinner dinner = dinnerRepository.GetDinner(5);

dinner.Country = "USA";
dinner.ContactPhone = "425-555-BOGUS";

if (!dinner.IsValid) {

    var errors = dinner.GetRuleViolations();

    // do something to fix the errors
}
```

If we attempt to save a Dinner in an invalid state, an exception will be raised when we call the Save() method on the DinnerRepository. This occurs because LINQ to SQL automatically calls our Dinner.OnValidate() partial method before it saves the Dinner's changes, and we added code to Dinner.OnValidate() to raise an exception if any rule violations exist in the Dinner. We can catch this exception and reactively retrieve a list of the violations to fix:

C#

```
Dinner dinner = dinnerRepository.GetDinner(5);

try {

    dinner.Country = "USA";
    dinner.ContactPhone = "425-555-BOGUS";

    dinnerRepository.Save();
}

catch {

    var errors = dinner.GetRuleViolations();

    // do something to fix errors
}
```

Because our validation and business rules are implemented within our model layer, and not within the UI layer, they will be applied and used across all scenarios within our application. We can later change or add business rules and have all code that works with our Dinner objects honor them.

Having the flexibility to change business rules in one place, without having these changes ripple throughout the application and UI logic, is a sign of a well-written application, and a benefit that an MVC framework helps encourage.

## Next Step

We've now got a model that we can use to both query and update our database.

Let's now add some controllers and views to the project that we can use to build an HTML UI experience around it.

[Previous](#)[Next](#)

# Studio User Guide

RELEASE: 2023.4 ▾

## TABLE OF CONTENTS

### [About Workflow Analyzer](#)

[Rules Naming Convention](#)

[Scope](#)

[Action](#)

[Managing Rules](#)

[Validation](#)

[Managing Errors](#)

[Command-Line Support](#)

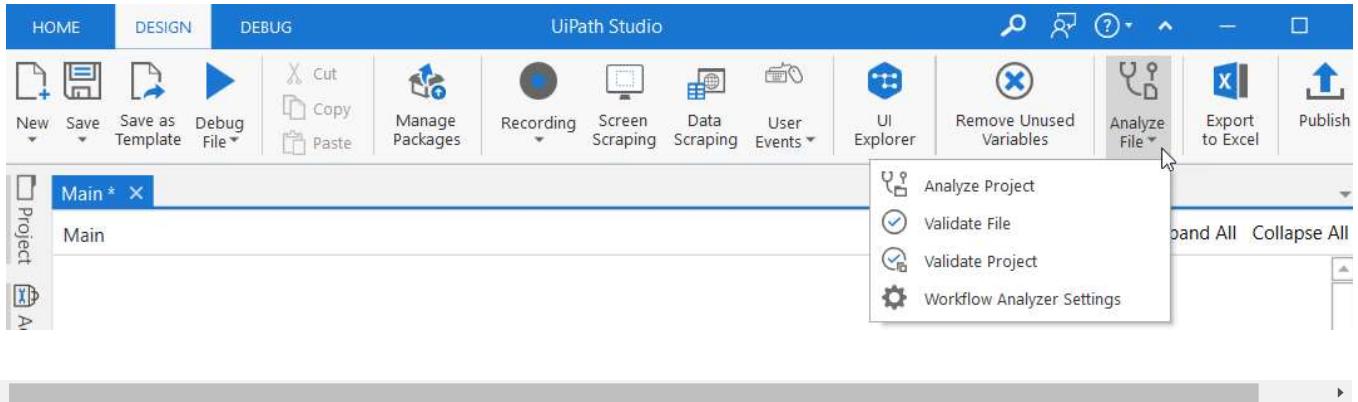
[Exporting Workflow Analyzer Results](#)

[Enforcing the Workflow Analyzer before Run, Publish, or Push/Check in](#)

# About Workflow Analyzer

**Workflow Analyzer** is a static code analyzer that ensures your project meets high quality and reliability standards. A static code analyzer checks for inconsistencies without actually executing the project, as opposed to dynamic analyzers which step in during execution.

**Workflow Analyzer** uses a set of rules to check for various inconsistencies unrelated to project execution. The rules are based on [Automation Best Practices](#) and take into consideration variable and argument naming, empty sequences or workflows, package restrictions, and so on. The analyzer does not identify errors in execution or compilation.



It is available in the **Design** ribbon tab, the **Analyze File** and **Analyze Project** buttons. The first performs an analysis on the file currently focused in the **Designer** panel, while the second analyzes all files in the automation project.

Studio comes with a set of built-in rules, identifiable by the ST- prefix. The [UIAutomation.Activities](#), [Excel.Activities](#), [Testing Activities](#), and [Mail.Activities](#) also have their own rules, identifiable by the prefix.

You can also create custom rules. For more information, see [Building Workflow Analyzer Rules](#) in the Developer guide.

Rules can also be configured when creating a Studio policy in Automation Ops. For more information, see the Automation Ops [guide](#).

#### **NOTE:**

Studio built-in rules cannot be ported to older Studio versions. However, rules from compatible activities packages can be used in older Studio versions that come with the Workflow Analyzer.

## Rules Naming Convention

Each rule has an ID, name, description, and recommendation. The ID contains the origin, category, and number. For example, [Variables Naming Convention](#) has the ST-NMG-001 ID:

- ST - reveals that the rule is built into Studio.
- NMG - shows that the rule is part of the [Naming Rules](#) category. Rules part of [Project Anatomy Rules](#) category have the ANA abbreviation, those part of [Design Best Practices](#) the DBP, and so on.
- 001 - is the rule number.

	Code	Name	Scope	Default action
<input checked="" type="checkbox"/>	ST-NMG-001	Variables Naming Convention	Activity	Warning

## Scope

Each rule has a scope to which it applies:

- **Activity:** The rules are enforced at activity level, checking variables, arguments, properties. [Variables Naming Convention](#) is one such rule.
- **Workflow:** Rules perform checks in a single project file, for example [Unused Variables](#).
- **Project:** Checks are performed at project level.

## Action

Rules have a number of actions that can be set:

- **Error:** Generates an error in the [Error List](#) panel.

- **Warning:** Generates a warning in the **Error List** panel.
- **Info:** Generates a message in the **Error List** panel.
- **Verbose:** Creates large log files.

Check the [Logging Levels](#) page to learn more about logging with Studio.

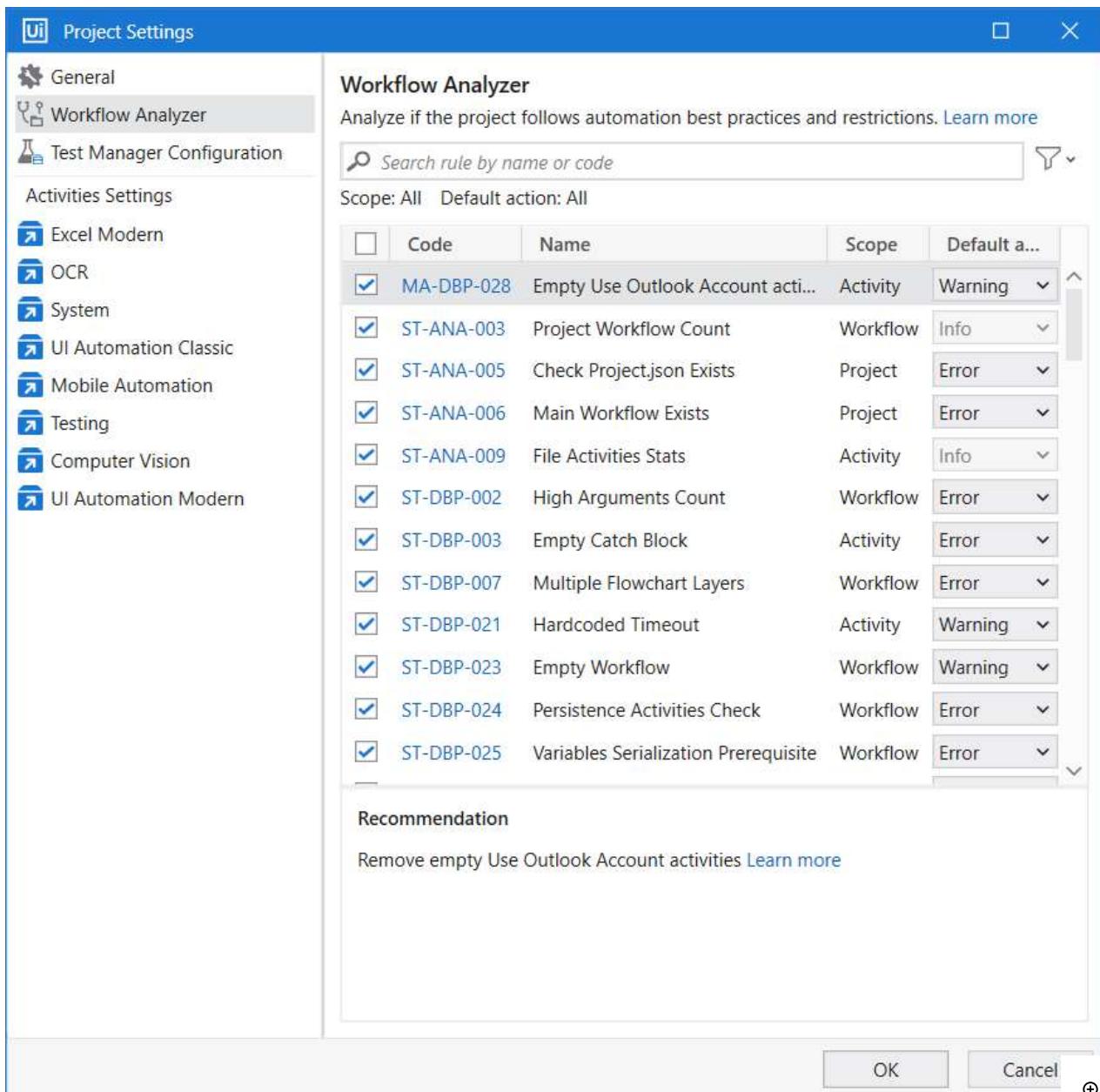
## Managing Rules

The project or file is analyzed based on a set of rules available in the **Project Settings** window:

- In the **Project** panel, click the icon, then select **Workflow Analyzer**.
- In the ribbon, click the **Analyze File** button, then select **Workflow Analyzer Settings** from the dropdown.

By default, rules are arranged by code in the window, and enabled rules are displayed first.

Use the search box at the top of the window to search rules by name or code. To filter rules by scope and default action, click the button to the right of the search box.



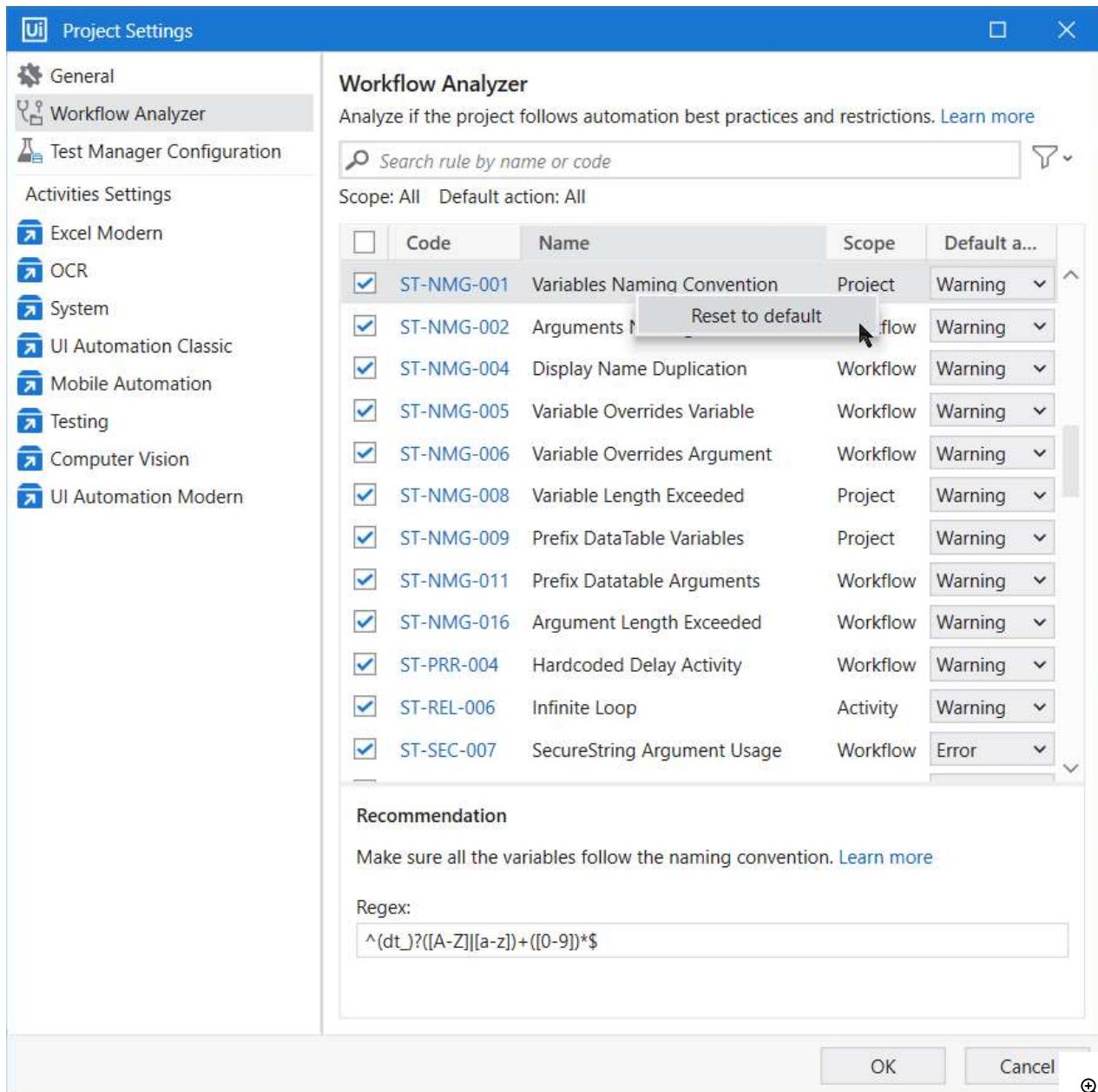
## Configure Rules

You can enable/disable a rule using the checkbox on its left, or change the default action using the dropdown on its right.

Some rules require no additional configuration, while others contain parameters that you can configure, such as thresholds not to be exceeded, or lists of specific items that are allowed or prohibited. Please note that when you configure rules that contain text fields (for example [SecureString Misusage](#) or [Package Restrictions](#)), text shouldn't be entered between quotes.

## Reset to Default

After any changes were made to the default values of rules, be it Regex or thresholds, the values may be reverted to default by right-clicking the rule and then selecting **Reset to Default**.



## Validation

Validation of the file or project is performed whenever the **Workflow Analyzer** is triggered. Validation options are available in the **Analyze File** ribbon button, **Validate File** and **Validate Project**.

This action checks whether variables, arguments, expressions, and imports are properly configured. The **Validate File** action can be triggered using the **F8** shortcut, while **Validate Project** using **Shift + F8**. Validation can be cancelled at any time.

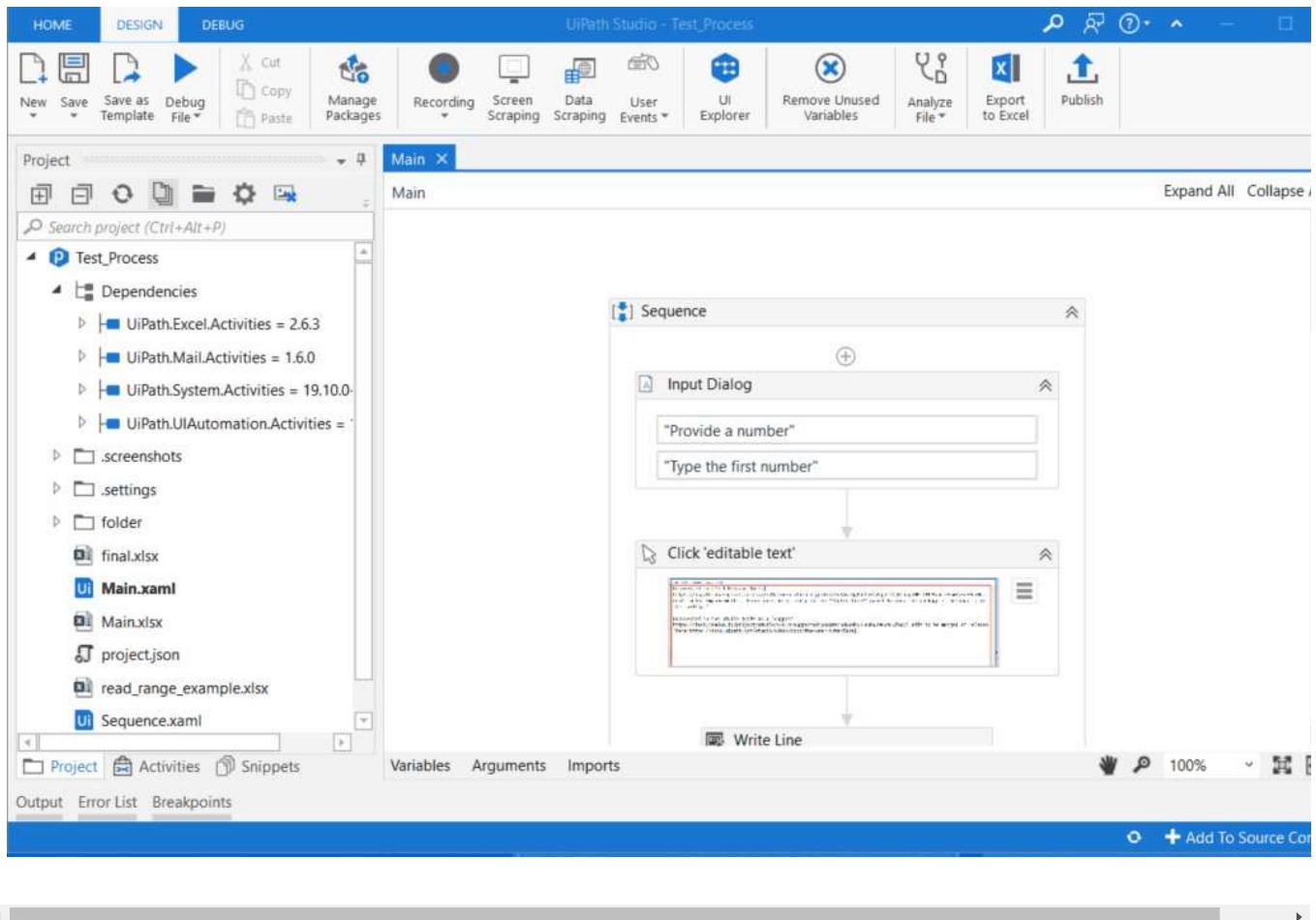
The file(s) that contain errors are marked in the **Project** panel with a red dot, and the errors are shown in the **Error List** and **Output** panels.

Double-clicking an item in the **Error List** panel opens the .xaml and highlights the element which threw the error, whether it's an activity, argument, or variable.

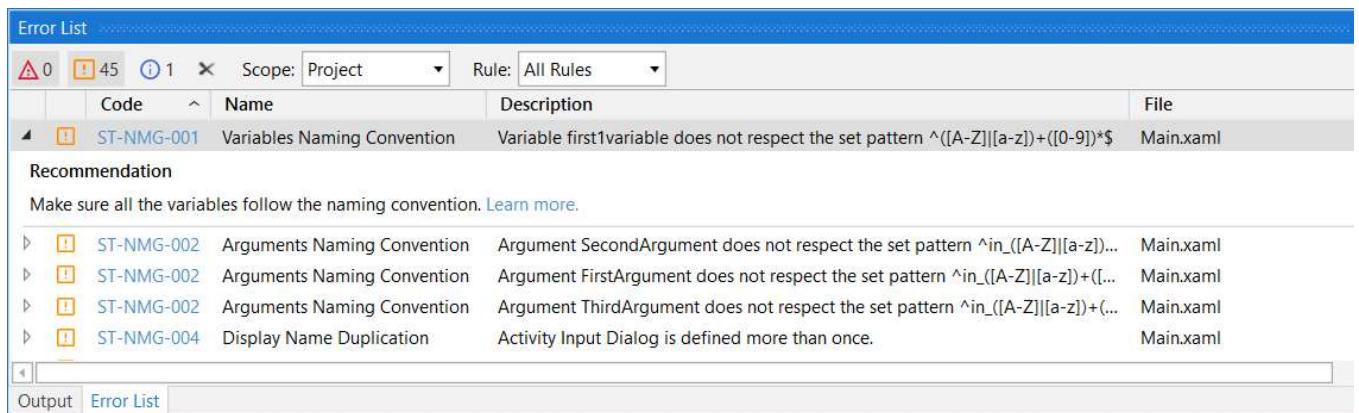
Unless all .xaml files are validated, the project cannot be debugged, executed or published. Breakpoint conditions are not evaluated.

## Managing Errors

When triggered, the **Workflow Analyzer** uses the configured ruleset to check the project or file, and logs the found errors in the **Error List** panel, in accordance with the rule action.



The **Error List** panel filters items by **Errors**, **Warnings** and **Messages**. The scope drop-down menu is useful for filtering errors by project, current file or a specific file in the automation. In addition, items can be filtered by rule ID.



Click a found warning or error to display the rule's recommendation and a link to the documentation page for each rule.

In addition to rule violations, the Error List panel also displays errors that may occur in the execution of workflow analysis. When the Workflow Analyzer or one of the configured rules fails to run, an error is logged with details about the cause of the error in the description.

# Command-Line Support

The UiPath.Studio.CommandLine.exe command-line user interface contains a set of parameters for checking files or projects against certain rules, even in CI/CD pipeline configurations.

UiPath.Studio.CommandLine.exe is available in the installation folder:

- For per-machine installations, the default path is %ProgramFiles%\UiPath\Studio\
- For per-user installations, the default path is %LocalAppData%\Programs\UiPath\Studio

## Configure Rules

By default, all Workflow Analyzer rules are enabled when installing Studio with the exception of the following:

- [Argument Default Values](#)
- [Undefined Output Properties](#)

You can configure rules:

- From the Workflow Analyzer Settings window, enable or disable rules by clicking the checkbox next to each one of them,
- From the RuleConfig.json file, path %LocalAppData%\UiPath\Rules. Find a specific rule, modify its parameter, change the `IsEnabled` parameter to `false` to disable the rule. Optionally, if the path is not accessible, you can place the RuleConfig.json file in a custom location and include the path to the file in the command. For more information, see [Analyze Files and Projects](#).

Rules are organized according to their unique ID, visible in the **Workflow Analyzer Settings** also.

For example, in the image below only the [Variables Naming Convention](#) rule was enabled and received the `([A-Z])` Regex expression. This means that variable names must be uppercase. All the other rules were disabled in this example.

```
"Rules": [
    {
        "RuleId": "ST-NMG-001",
        "IsEnabled": null,
        "DefaultIsEnabled": true,
        "DefaultErrorLevel": "Warning",
        "Parameters": [
            {
                "Name": "Regex",
                "Value": "([A-Z])"
            }
        ],
        "ErrorLevel": null
    },
    {
        "RuleId": "ST-NMG-008",
        "IsEnabled": null,
        "DefaultIsEnabled": false,
        "DefaultErrorLevel": "Warning",
        "Parameters": [
            {
                "Name": "Length",
                "Value": null
            }
        ],
        "ErrorLevel": null
    },
    {
        "RuleId": "TA-DBP-003",
        "IsEnabled": null,
        "DefaultIsEnabled": false,
        "DefaultErrorLevel": "Info",
        "Parameters": [],
        "ErrorLevel": null
    }
]
```

The following parameters are available for each rule:

Parameter	Description
RuleId	The ID of the rule.
.IsEnabled	Whether the rule is enabled or disabled. If set to null, the value of the parameter DefaultIsEnabled is applied.
DefaultIsEnabled	Whether the rule is enabled or disabled by default.
DefaultErrorLevel	The default level of the log message (Error, Warning, Info, Verbose).
Parameters	Additional parameters that the rule can contain (for example, Name or Value).
ErrorLevel	The level of the log message. If set to null, the value of the parameter DefaultErrorLevel is applied.

## Analyze Files and Projects

After configuring the rules, access the `UiPath.Studio.CommandLine.exe` command-line user interface.

The following Workflow Analyzer commands are available:

- `analyze` - Analyzes the whole project. Requires the path to the `project.json` file.
- `analyze-file` - Analyzes a single file. Requires the path to the `.xaml` file.

The following arguments are available for Workflow Analyzer commands:

Argument	Description
<code>-p, --project-path</code>	For analyzing one file, provide the path to the <code>.xaml</code> . For analyzing the entire project, specify the path to the <code>project.json</code> file.
<code>-c, --config-path</code>	Path to the <code>RuleConfig.json</code> file. Specify this path only if the rule configuration file is not placed in the default location ( <code>%LocalAppData%\UiPath\Rules</code> ).
<code>--help</code>	View the arguments available for each command.
<code>--version</code>	Check the version of <code>UiPath.Studio.CommandLine.exe</code> .

For example, the command `UiPath.Studio.CommandLine.exe analyze-file -p "C:\BlankProcess\Main.xaml"` analyzes only one file in a project, `Main.xaml`. The output of the command returns a json-encoded dictionary with the following information for each rule violation:

- `FilePath`: The path to the analyzed file.
- `ErrorCode`: The rule ID.
- `ActivityDisplayName`: The display name of the activity (null if not applicable).
- `Item` - The name and type of the item (for example, activity or variable) where the error message was generated.
- `ErrorSeverity`: The value of the `DefaultAction` parameter, which is the configured log message for each rule, either `Warning`, `Error`, `Info` or `Verbose`.
- `Description`: The rule's description.
- `Recommendation and URL`: The recommended changes for solving the issue, together with the documentation link with more information.

```
C:\Program Files (x86)\UiPath\Studio>UiPath.Studio.CommandLine.exe analyze-file -p "C:\BlankProcess\Main.xaml"
#json{
    "d9ae4273-56aa-43bf-bc58-828fea0656b7-FilePath": "C:\\\\BlankProcess\\\\Main.xaml",
    "d9ae4273-56aa-43bf-bc58-828fea0656b7-ErrorCode": "ST-NMG-002",
    "d9ae4273-56aa-43bf-bc58-828fea0656b7-ActivityDisplayName": null,
    "d9ae4273-56aa-43bf-bc58-828fea0656b7-Item": "argument1, of type Argument",
    "d9ae4273-56aa-43bf-bc58-828fea0656b7-ErrorSeverity": "Warning",
    "d9ae4273-56aa-43bf-bc58-828fea0656b7-Description": "Argument argument1 does not respect the set pattern dt_)?([A-Z]|[a-z])+([0-9])*$",
    "d9ae4273-56aa-43bf-bc58-828fea0656b7-Recommendation": "Make sure all the arguments follow the naming convention. [Learn more.](https://docs.uipath.com/studio/lang-en_US/v2020.10/docs/st-nmg-002)",
}
```

To use a RuleConfig.json file placed in a custom location, add the location to the command. Using the previous command as an example, to configure rules with the file located at C:\CustomFolder\RuleConfig.json, run the command as follows: UiPath.Studio.CommandLine.exe analyze-file -c "C:\CustomFolder\RuleConfig.json" -p "C:\BlankProcess\Main.xaml".

To return the exit code of the command, execute echo %errorlevel% after you run it. If there are no messages with the ErrorSeverity Error, the code 0 is returned. Otherwise, 1 is returned.

Please note that some of the entries may not be associated with a rule, but can be errors generated by the Workflow Analyzer. For example, in the case of an empty .xaml file, an error is displayed.

The Globally Unique Identifiers (GUID) used as prefixes for an entry are generated on each run and apply to the current result only.

## Exporting Workflow Analyzer Results

To configure Studio to export the results of each workflow analysis to the project folder, go to **Studio Backstage View > Settings > Design** and enable the **Export Analyzer results** option.

When this option is enabled, the results of each workflow analysis are saved in the \.local\analysis\ subfolder of the project folder in a file named with the timestamp of the analysis followed either by project\_analysis\_results.json (when the **Analyze Project** option is used) or file\_analysis\_results.json (when the **Analyze File** option is used).

**NOTE:**

The .local folder is hidden. You can enable viewing hidden items from the Windows File Explorer settings.

The files contain the following information about each enabled rule:

- RuleId - The rule ID.
- RuleName - The name of the rule.
- Parameters - The customizable parameters in the rule, if applicable.
- Severity - The rule action, if the rule is violated.
- ErrorsDescription - A list of error messages generated by the rule, if applicable.

## Enforcing the Workflow Analyzer before Run, Publish, or Push/Check in

You can prevent executing, publishing, and pushing/checking in to remote repositories projects that contain Workflow Analyzer errors by enabling the following options from **Studio Backstage View > Settings > Design**:

- **Enforce Analyzer before Run** - Whenever running/debugging a file or project is initiated, the Workflow Analyzer checks all the rules with the Error action and execution is allowed only if no errors are found.
- **Enforce Analyzer before Publish** - Whenever publishing is initiated, the Workflow Analyzer checks all enabled rules regardless of their action and publishing is allowed only if there are no rule violations with the action Error.
- **Enforce Analyzer before Push/Check-in** - Whenever sending a project to a remote repository is initiated (**Commit and Push** for GIT, **Check in** for SVN and TFS), the Workflow Analyzer checks all enabled rules regardless of their action and the operation is allowed only if there are

no rule violations with the action Error.

When Enforce Analyzer before Publish is enabled, if publishing is successful (there are no rule violations with the action Error) the results of the workflow analysis are included in the published .nupkg package in the file project\_analysis\_results.json located in \lib\net45\analysis\. The file contains the following information about each enabled rule:

- RuleId - The rule ID.
- RuleName - The name of the rule.
- Parameters - The customizable parameters in the rule, if applicable.
- Severity - The rule action (if the rule is violated).
- ErrorsDescription - A list of error messages generated by the rule, if applicable.



Default Theme

English

# UI Automation Activities

## TABLE OF CONTENTS

---

[Workflow Analyzer rules](#)

## Workflow Analyzer rules

The Workflow Analyzer rules section includes the following pages:

- [Classic Workflow Analyzer rules](#)
- [Modern Workflow Analyzer rules](#)



Default Theme

English

# Studio User Guide

RELEASE: 2023.4

## TABLE OF CONTENTS

### [Managing Projects With GIT](#)

[Authenticating to GIT](#)

[Over HTTPS](#)

[Over SSH](#)

[Authentication Failed](#)

[Cloning a Remote GIT Repository](#)

[Adding a Project to GIT](#)

[Committing and Pushing to GIT](#)

[Changing the Last Commit](#)

[Undoing Pending Changes](#)

[Copying a Project to GIT](#)

[Creating and Managing Branches](#)

[Viewing the Commit History](#)

[Comparing Two Versions of a File](#)

[Creating a Branch from a Previous Commit](#)

[Solving Conflicts](#)

[Disconnecting From GIT](#)

[Changing the Signature](#)

[Changing the Credentials](#)

[Using GIT With a Proxy Server](#)

# Managing Projects With GIT

GIT integration in Studio requires the Microsoft Visual C++ Redistributable for Visual Studio 2015, 2017, 2019, and 2022. Check the [Software Requirements](#) page.

# Authenticating to GIT

Authentication methods in Studio differ in accordance with the methods used for cloning a GIT repository, either HTTPS or SSH. Check this [page](#) to see which you should use if you're working with GitHub.

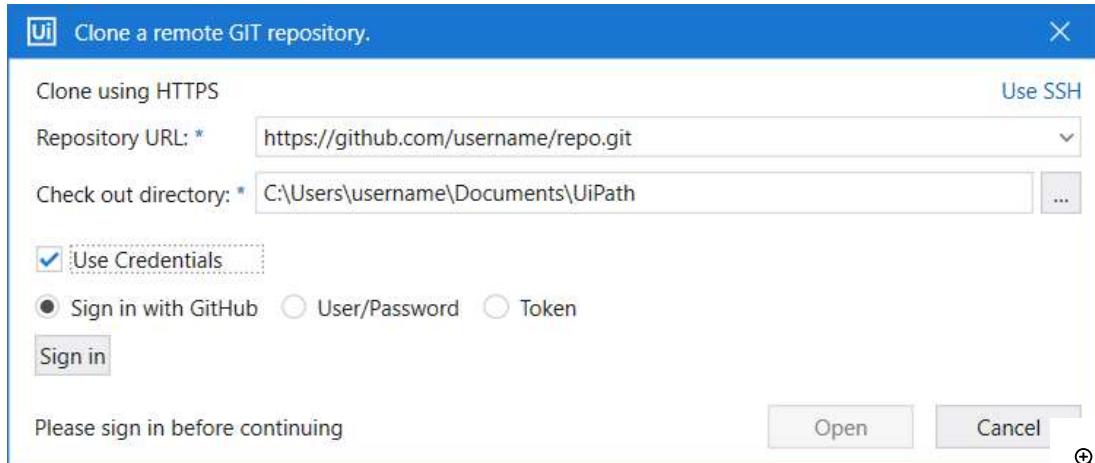
## NOTE:

- The GIT credentials you provide in Studio are stored in the Windows Credential Manager.
- The GIT integration with Studio currently supports two-factor authentication only for GitHub if you authenticate by signing in to the service. For other tools, use the SSO authentication method with a personal token, or the basic access authentication method.

The current guide details the steps for authenticating to a GitHub repository, but the Git integration in Studio is not limited to just this service.

## Over HTTPS

When cloning a remote GIT repository or copying the current project to an existing GIT repository using HTTPS for the first time, you must provide your GIT credentials. These credentials must be entered in the **Use Credentials** fields:



You can authenticate using the following options:

- Sign in with GitHub** - Sign in with your GitHub account.
- User/Password** - Enter your user and password.
- Token** - Enter your user and personal access token.

Follow the steps detailed in this [page](#) to generate a GIT token for your GitHub repository.

## IMPORTANT:

The **Sign in with GitHub** option is available only for repositories hosted on [github.com](#) and requires the [UiPath GitHub App](#) to be installed in your organization or account.

## Over SSH

When cloning a repository or copying the current project to an existing GIT repository using SSH for the first time, you have the option of using a private key:

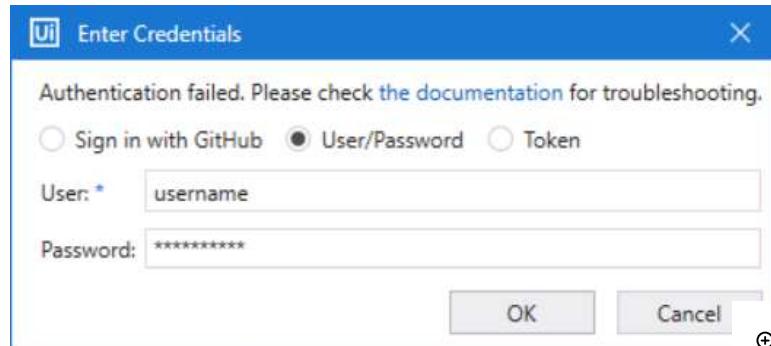


Add the **Private Key Path** and the **Password**, and then click **Open** to clone your remote GIT repository. Check out the steps detailed [here](#) to generate a SSH key for your GitHub repository.

## Authentication Failed

### Over HTTPS

When cloning a GIT repository, the **Enter Credentials** window is displayed with the message **Authentication failed** if the provided credentials were incorrect:



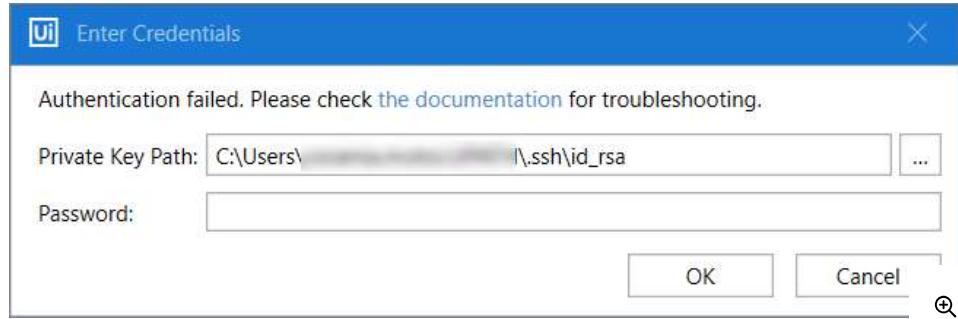
Enter the correct credentials and try again. You can also use Git Bash, for example, to clone your repository remotely and thus, check if the username and password are correct.

In the following image, we tried to clone a repository over HTTPS, but entered an incorrect password. The Git Bash window shows that the credentials were incorrect.

```
MINGW64:/c/Users/ [REDACTED]
$ git clone https://github.com/... /DocTests.git
Cloning into 'DocTests'...
Logon failed, use ctrl+c to cancel basic credential prompt.
remote: Invalid username or password.
fatal: Authentication failed for 'https://github.com/... /DocTests.git/'
```

### Over SSH

The following **Enter Credentials** window is displayed when authentication fails over SSH:



Enter the correct **Private Key Path** and **Password** and try again. You can also use Git Bash, for example, to clone your repository remotely and thus, check if the token and/or password are correct.

In the following image, we tried to clone a repository over SSH, but we didn't have any public SSH keys set up on our account.

```
MINGW64:/c/Users/...
$ git clone git@github.com: /DocTests.git
Cloning into 'DocTests'...
git@github.com: Permission denied (publickey).
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.
```

## Cloning a Remote GIT Repository

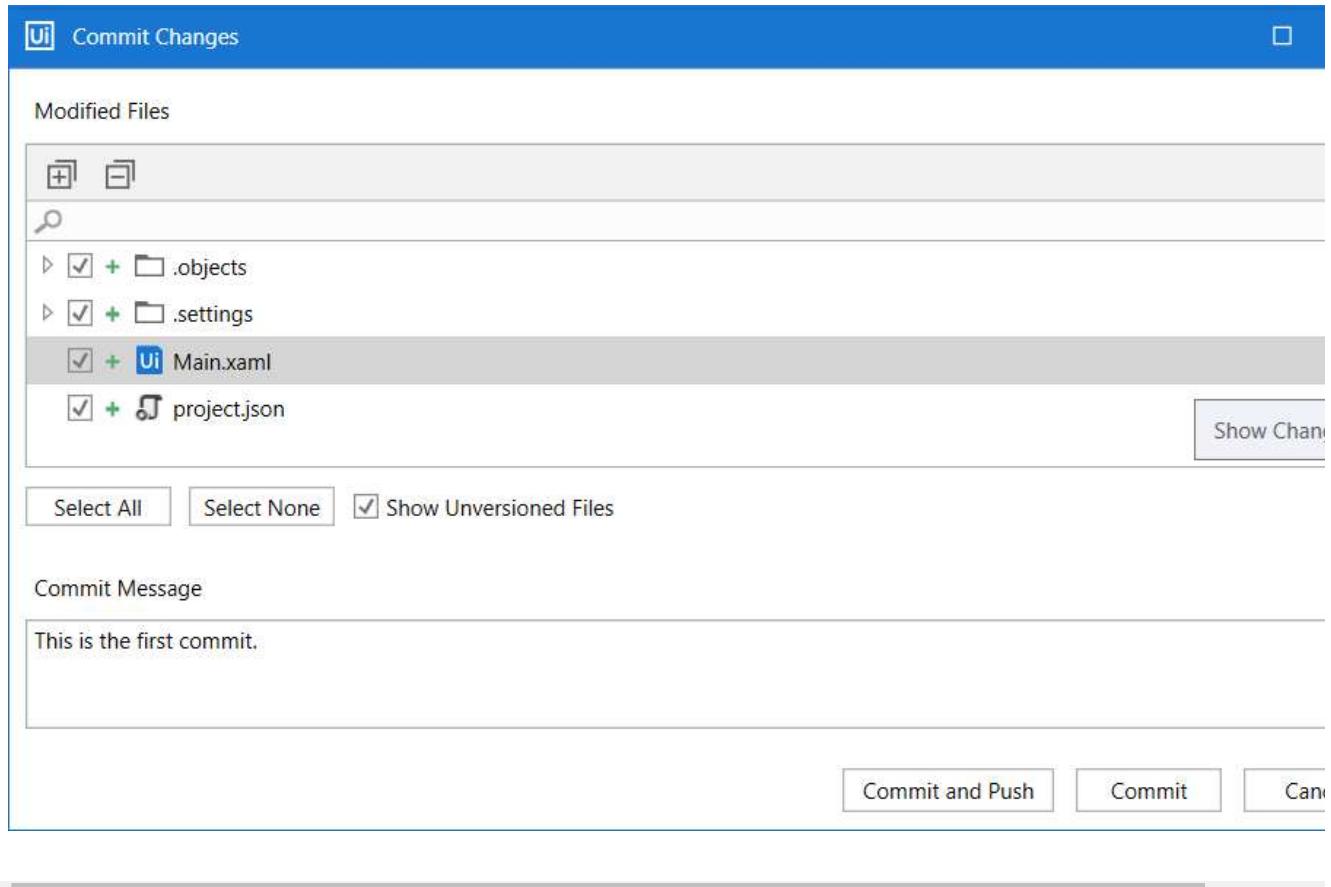
1. In the **Team** tab, select **Clone Repository**. The **Clone a remote GIT repository** window is displayed.
2. Select either **Use HTTPS** or **Use SSH**.
3. Type in the **Repository URL**, and choose an empty **Check out directory**.
4. Select **Use Credentials / Use Key** and configure authentication (either sign in with GitHub, enter user and password, enter user and token for HTTPS, or enter private key path and password for SSH).
5. Click **Open**, Studio opens the project in the **Designer** panel.
6. In the **Open** window, select a project.json file to open in Studio.

After cloning a GIT repository to a local working directory, the .git subdirectory is created containing the necessary GIT metadata. The metadata includes subdirectories for objects, refs, and template files. In addition, a HEAD file is also created, which points to the currently checked out commit.

## Adding a Project to GIT

The **GIT Init** feature adds the current project to a local GIT repository. Access the command from the **Team** tab, or the status bar.

1. Create or open a project in Studio. Click the **Start** tab > **Team**. The **Team** tab is displayed.
2. Click the **GIT Init** button, and then select a path where the repository should be initialized. The location may be the same as the project or the parent folder. The **Commit changes** window opens.



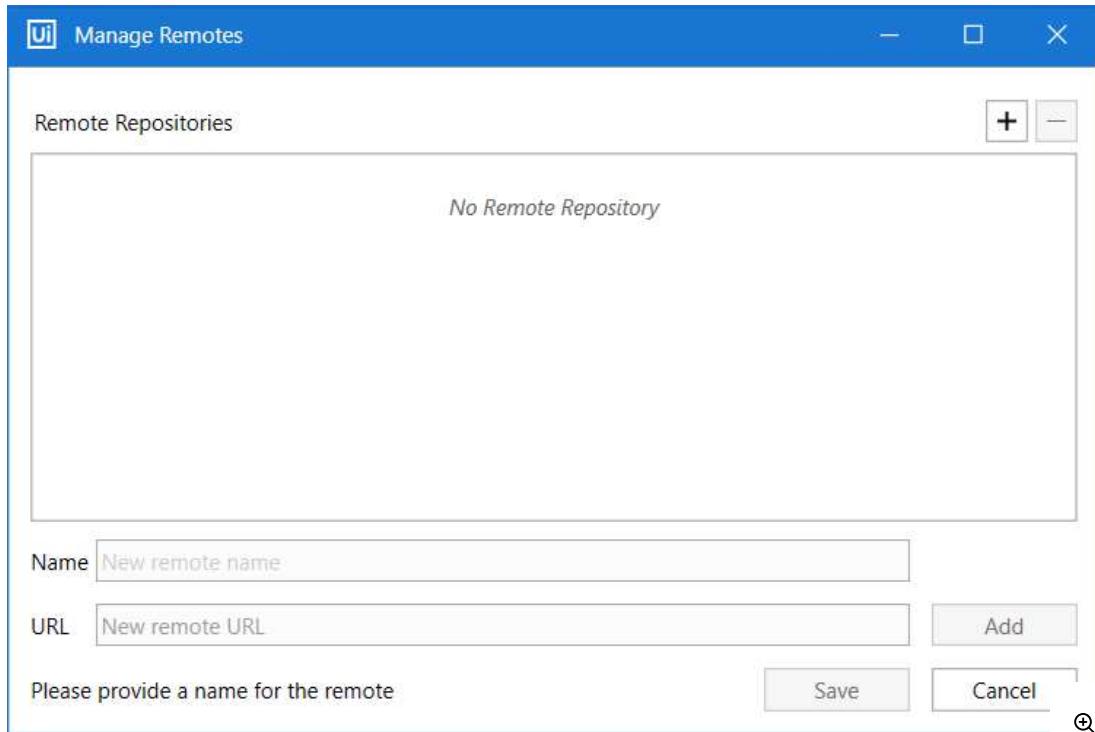
3. The **Modified Files** section shows the project's files that are to be added to the Git repo. Clear the box next to the ones that you don't want to add or use **Select All**, **Select None**.
4. Select the **Show Unversioned Files** box to add unversioned files to the list.

Write a **Commit Message**. Click the **Commit** button to commit the changes to the local Git repository.

When a project is added to GIT, the context menu in the **Project** panel includes GIT-specific options. For more information, see [Context Menu Options for GIT](#).

## Committing and Pushing to GIT

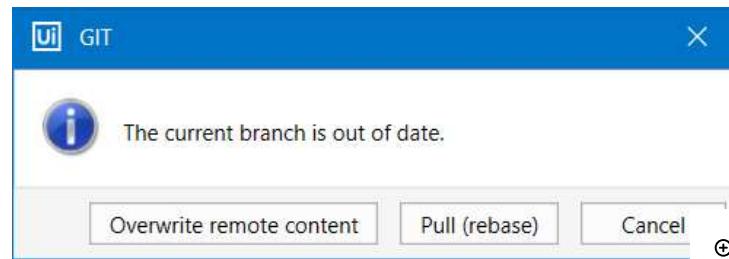
1. From the same **Commit Changes** window, click the **Commit and Push** button to commit the changes and push them to the remote repository. This **Manage Remotes** window is displayed. The window is also available from the status bar.



2. In the **Name** section, add the name of the remote repository.

3. In the **URL** section, add the remote URL.

If you want to make modifications to the added repositories, simply click an entry, change the name and URL, then click the **Update** button. When you're done click **Add**, then **Save**. The following message box opens. This means that the local repository is not synchronized with the remote one.



- Click the **Overwrite remote content** button to push the local versions of files to the remote repository and overwrite the files there.
- Click the **Pull (rebase)** button to pull the remote files and rebase the current branch.
- Click the **Cancel** button to discard the whole operation.

The number of unpushed changes, and newly added files are visible in the status bar. Click the icon to open the **Commit Changes** window, or the icon to push changes.

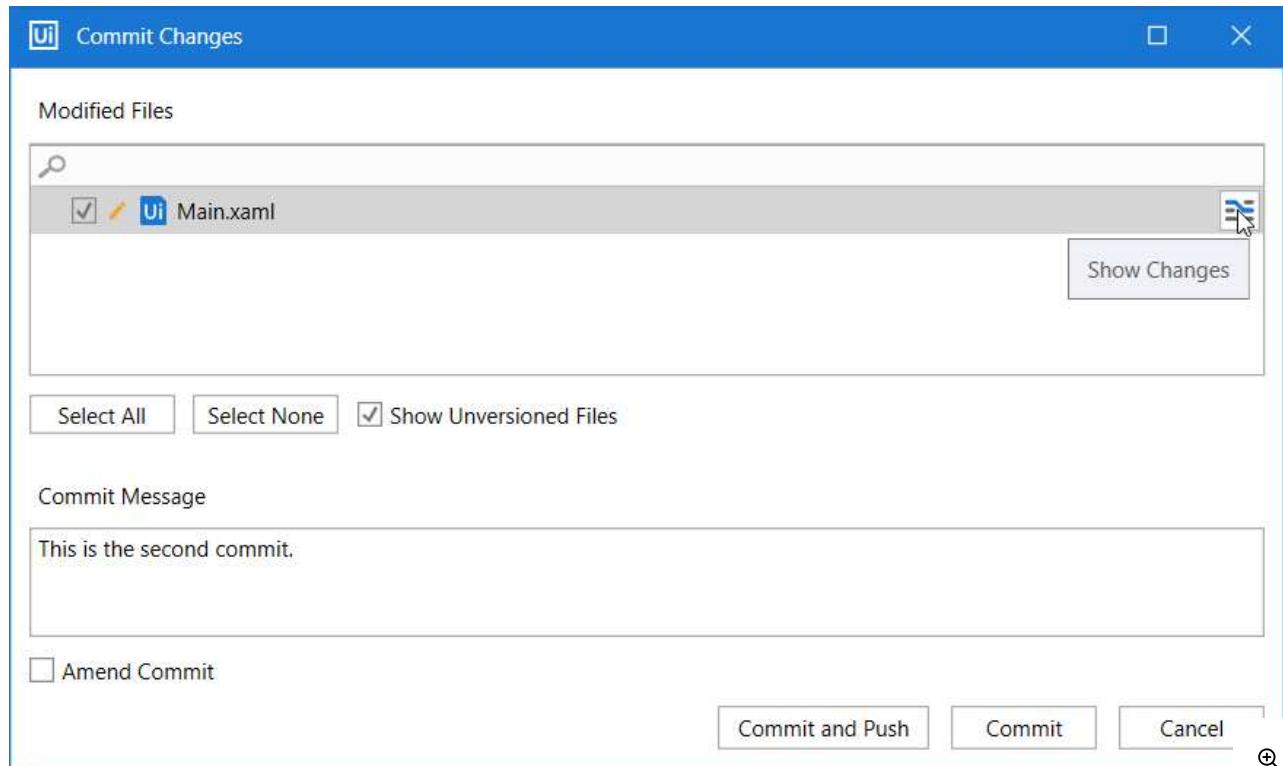
#### **NOTE:**

If you edit a file from a project added to source control in an external editor, the change is visible in the **Project** panel and the status bar only after you click **Refresh**

## Changing the Last Commit

Studio integration with Git also comes with an **Amend Commit** option for changing the last performed commit, before the push was performed.

1. Right-click a modified file in the **Project** panel and select **Commit**. The **Commit Changes** window is displayed.

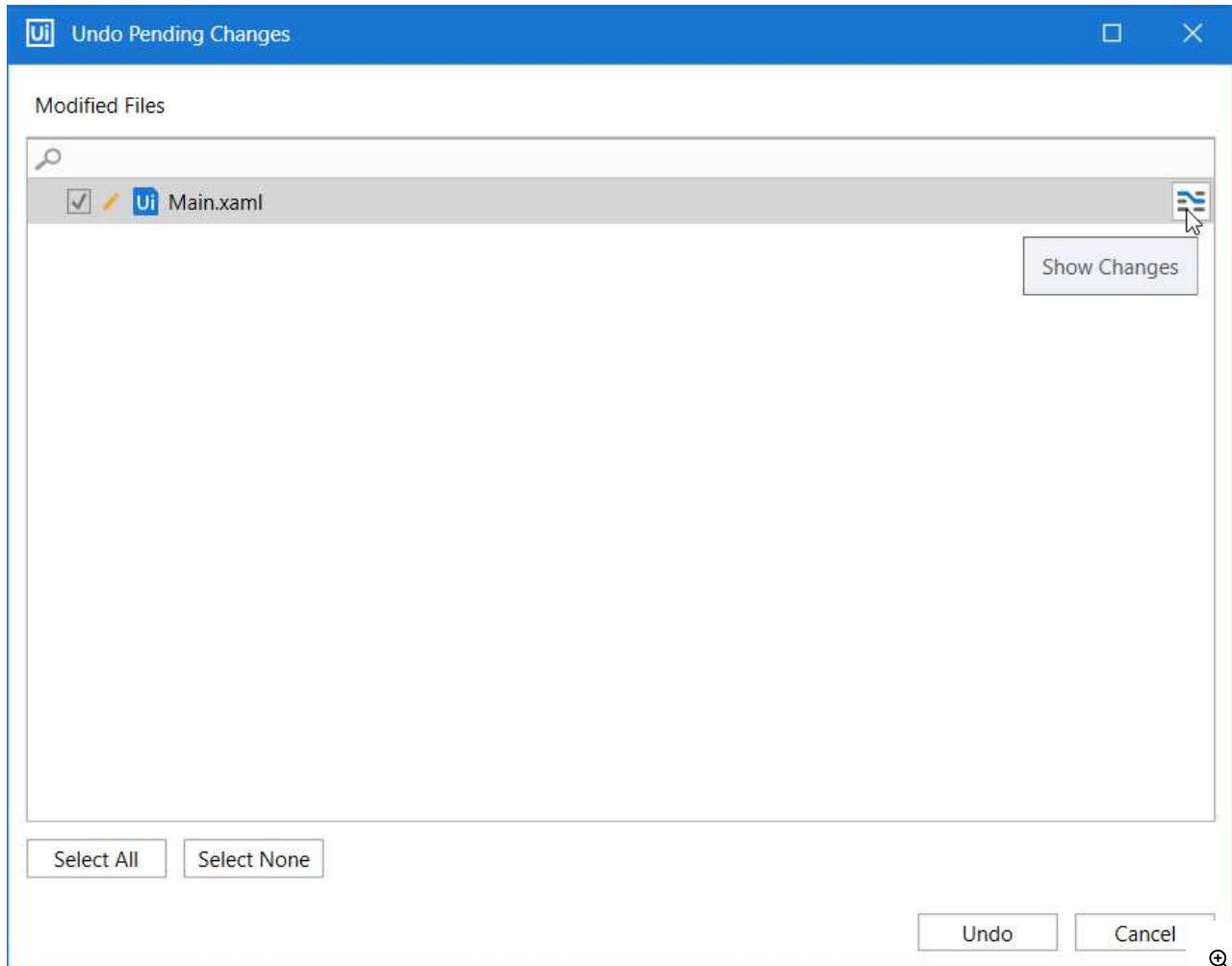


2. Select the **Amend Commit** box. The last commit message is displayed, together with the files that were committed. To view changes between the current file and the last commit, use the **Show Changes** option.
3. Change the commit message and select the files that you would like to include. Click the **Commit and Push** or **Commit** button.

## Undoing Pending Changes

Studio comes with the option to undo changes that have been made to versioned files, before you commit and push them to the remote repository.

After making changes to a file in the local repository, click **Undo** in the GIT context menu to open the **Undo Pending Changes** window.



Select the checkbox next to the files and click **Undo**. The files are now reverted to the state before the changes were made.

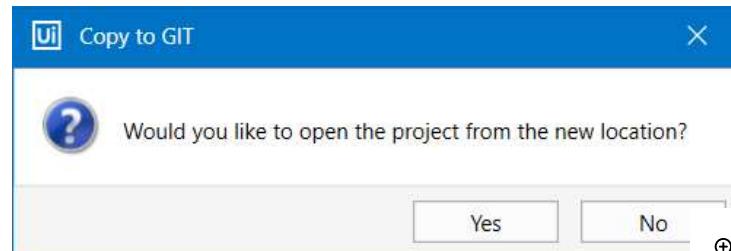
**NOTE:**

The **Undo** does not cover unversioned files. If you create new files and then select **Undo**, the files are not removed from the project. Once added to the project tree, new files remain there unless they are manually deleted.

## Copying a Project to GIT

The **Copy to GIT** button in Studio Backstage view and status bar allows you to copy the current project to an existing GIT repository.

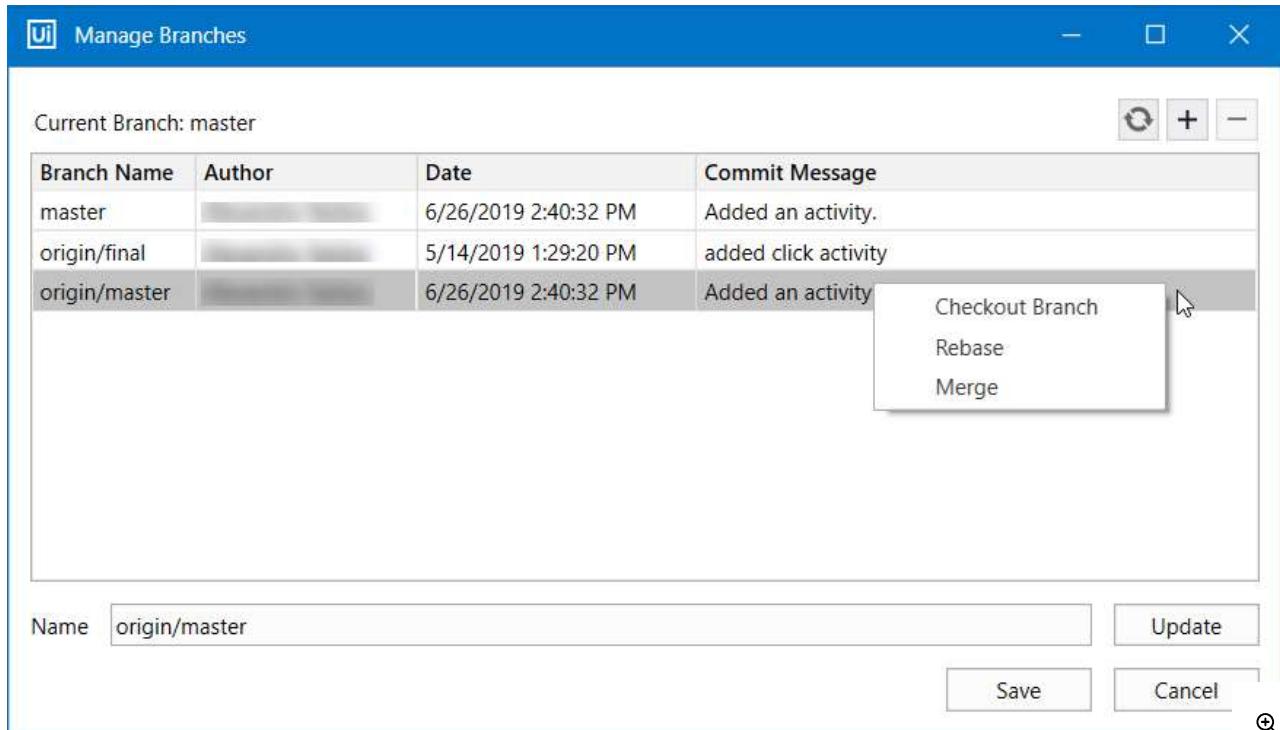
To do so first open or create a project in Studio. In the **Team** tab, select **Copy to GIT** and pick an existing GIT repository folder on your machine. The project is added to the local GIT repository and the **Copy to GIT** message box opens.



- Select **Yes** to open the project from the new location. The **Commit Changes** window opens. Write a **Commit message** and click **Commit and Push** or just **Commit**.
- Select **No** to return to the Studio user-interface.

## Creating and Managing Branches

Add and manage branches from the **Manage Branches** window. To access it, either right-click the project node or a file in the **Project** panel and select **Manage Branches**, or use the  branch menu in the status bar.



- To add a branch:

1. Select a branch in the table, and then click **Plus**  at the top of the window.
2. Enter a name for the branch, click **Create branch from branch\_name**, and then click **Save**.

The branch is added to the list.

- To refresh the list of branches, click **Refresh** 
- To delete a local branch, select it, and then click **Delete** 
- To manage branches, right-click any branch and select one of the options from the menu:
  - The **Checkout branch** option switches to the selected branch.
  - The **Rebase** option rebases the current branch onto the selected branch.
  - The **Merge** option merges the selected branch into the current branch.

To merge a branch into the master of a GIT remote repository, you need to have the master branch checked out in Studio and then merge the modified branch into master.

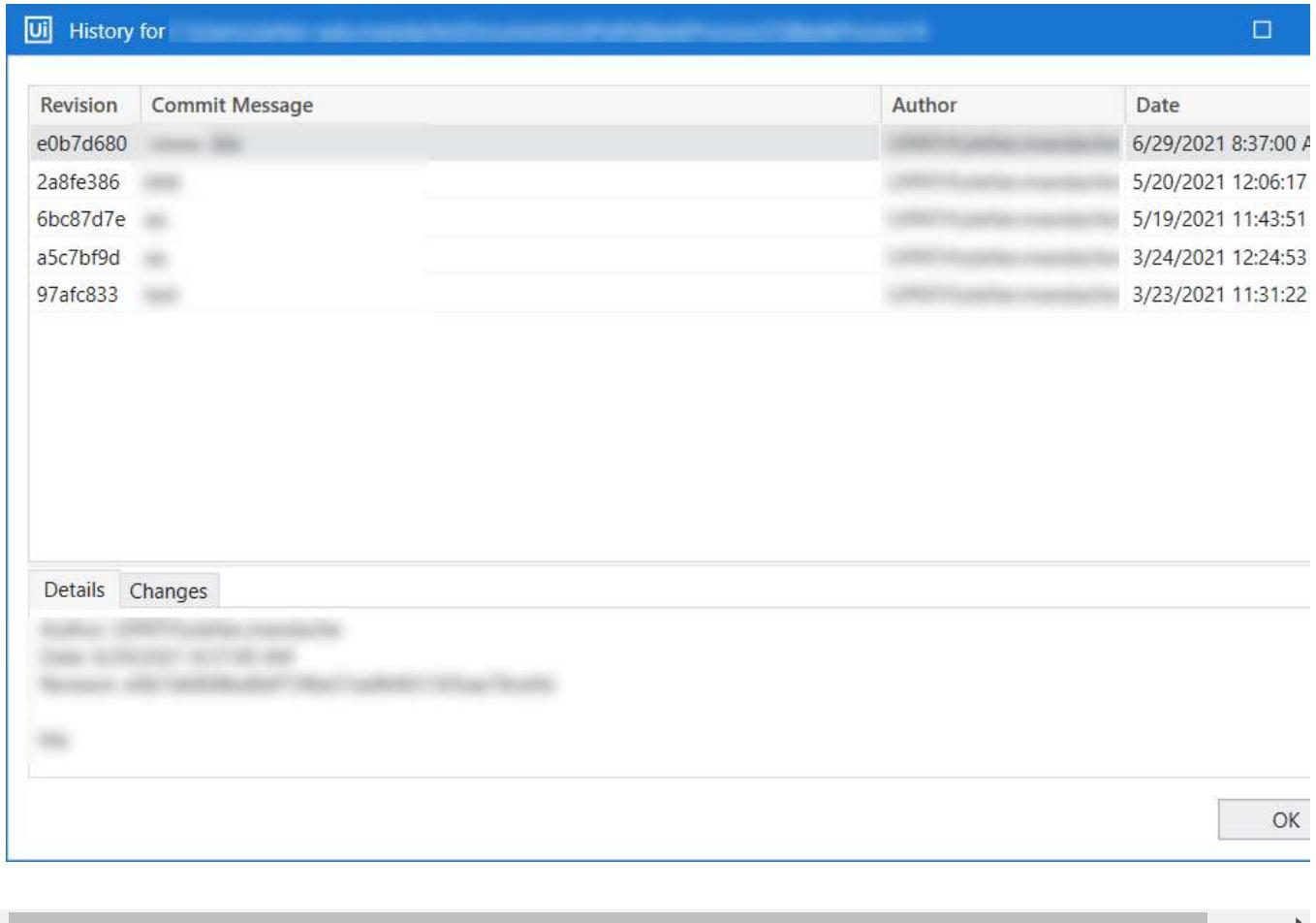
- To switch between branches, open the  branch menu in the status bar, and then select the branch to switch to from the list of recently checked out branches.

**NOTE:**

Branches need to be added locally in Studio from the **Manage Branches** window. Adding branches from the remote GIT repository may cause issues when trying to switch between branches.

## Viewing the Commit History

To view the commit history for a project or for a specific file or folder in a project, right-click the project node, a file, or folder in the **Project** panel, and then select **Show History**. This opens the **History** window which displays a list of existing revisions for the selected file, folder, or project. For each commit, the commit hash, message, author, and date are displayed in a table on the upper part of the window. You can view more information about a selected commit in the **Details** and **Changes** tabs on the lower part of the window.



## Comparing Two Versions of a File

To compare two versions of the same file:

- If you opened the history for a file, right-click a commit in the History window, and then select **Compare with Previous**, **Compare with Local**, or **Compare with Latest**.
- If you opened the history for a folder or project, select a commit in the History window, and then, in the Changes tab, double-click a file to compare it with its previous version.

## Creating a Branch from a Previous Commit

To create a branch from a specific commit:

- Right-click a commit in the History window, and then select **Create branch**.

2. In the **Create branch** window:

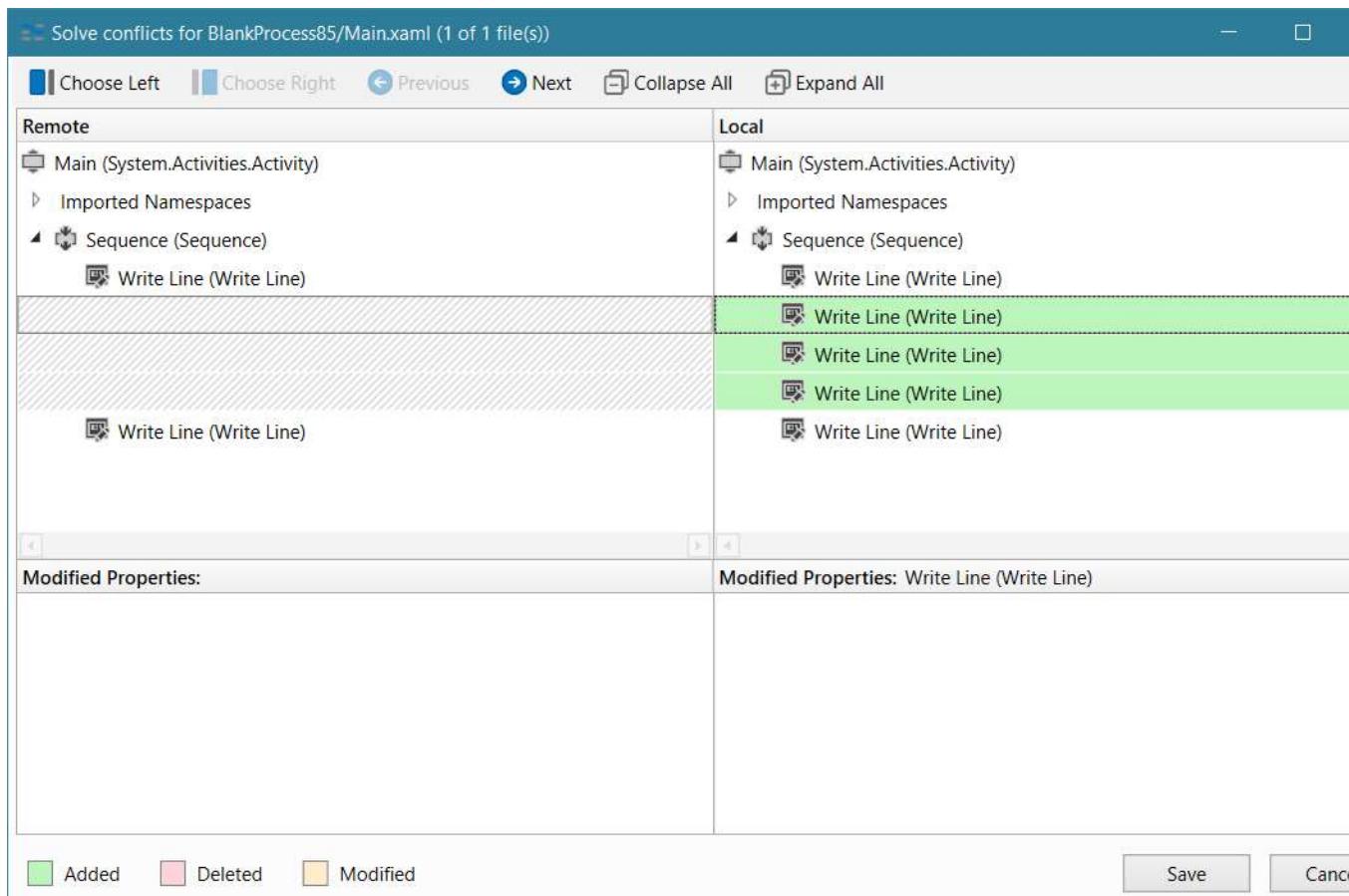
- Enter a name for the new branch.
- To also check out the new branch, make sure the **Checkout after create** option is selected.

3. Click **OK**.

## Solving Conflicts

GIT integration with Studio comes with a feature for solving conflicts that may occur when performing the **Rebase** or **Push** command, found in the **Commit Changes** window.

Whenever Studio detects a conflict between the local file and the one found in the remote repository, the **Solve conflicts** window is displayed.



The window is similar to [File Diff](#), showing the differences between the **Remote** version of the file and the **Local** version.

The following table describes the options available in the **Solve conflicts** window.

Option	Description
<b>Choose Left</b>	Select the left file representing the file in the remote repository to push.
<b>Choose Right</b>	Select the right file representing the file in the local repository to push.
<b>Save</b>	Click <b>Save</b> after choosing the left or right file.

Option	Description
<b>Cancel</b>	Cancel the operation and exit the <b>Solve conflicts</b> window.
<b>Previous</b>	Navigates to the previous change in the compared files.
<b>Next</b>	Navigates to the next change in the compared files.
<b>Collapse All</b>	Collapses all nodes in the .xaml files.
<b>Expand All</b>	Expands all nodes in the .xaml files.

## Disconnecting From GIT

The **Disconnect** option from Studio Backstage view > **Team** tab is available for versioned files in the following two cases:

1. A process is initialized as a local GIT repository. Create a new process, use **GIT Init** to add it to a local GIT repository and then use **Disconnect** to remove the subversion tag.
2. The subversion tag can be removed by clicking **Disconnect** for a GIT repository which includes parent and child projects.
3. If you disconnect a child project, then the entire GIT repository that contains the opened project is disconnected from source control. A message box is displayed in Studio requiring your confirmation before the disconnect action is performed.

## Changing the Signature

Changing the GIT commit signature can be done from the **Team** tab > **Change Signature**. Fill in your name and email address, and click **OK**.

## Changing the Credentials

To update the credentials used to connect to a remote repository:

1. From the  branch menu in the status bar, select **Manage Remotes**.
2. In the Manage Remotes dialog, right-click a remote repository and select **Change Credentials**.
3. Depending on how the repository was cloned (via HTTPS or SSH), either sign in to a GitHub account (for repositories hosted on github.com only), enter the username/password or the username/token, or point to a new private key and click **OK**.

The credentials are updated in the Windows Credential Manager.

## Using GIT With a Proxy Server

GIT integration in Studio supports accessing remote repositories if internet access is through a proxy server. This can be done in two ways: either configured at machine level in the **Proxy Settings** window or by making changes to git commands.

Proxy details configured in the **Proxy Settings** window are taken into account, without the need of entering them in the `.gitconfig` file.

To configure proxy details with git commands, add them to GIT configuration files in the following form:

```
[http "https://domain.com"] proxy = http://proxyUsername:proxyPassword@proxy.server.com:port
```

GIT configuration files can be found at the following locations:

- config file: %ProgramData%\Git
- .gitconfig file: %UserProfile%
- local config file from project level, for example %UserProfile%\Desktop\estproject\.git.



Default Theme

English

# Studio User Guide

RELEASE: 2020.10

## TABLE OF CONTENTS

### About Automation Projects

Setting the Project Language

C# Limitations

Setting the Project Version

Semantic Versioning

Legacy Versioning

Managing Projects

Context Menu for Projects

Adjusting Project Settings

About the Project.Json File

# About Automation Projects

UiPath Studio allows you to create two types of standalone automation projects: process or library. Processes can incorporate all types of workflows, sequence, flowchart, state machine and global exception handler, while the global exception handler isn't available for libraries. Moreover, [libraries](#) can be added as dependencies to automation processes.

When you start a new process in Studio, a folder is created with your custom name to the selected location. Projects are saved in the %USERPROFILE%\Documents\UiPath directory by default, unless this location is manually changed.

By default, the project folder includes the following files and subfolders:

- **Files**

- Main.xaml - Created by default to hold your main workflow. In addition, all the other automation XAML files you add to the project are stored in the project folder. Optionally, you can set a different file as main. All the files must be linked through the [Invoke Workflow File](#) activity to the file set as main or to a file marked as an entry point to the project. In the case of test projects created in Studio Pro, a TestCase.xaml file is created by default instead of Main.xaml.
- project.json - Contains information about your automation project.

- **Subfolders**

- .entities - Contains data about entities imported from the Data Service, if any are used in the project.

- `.local` - Contains data cached locally for the project.
- `.objects` - Contains data related to items added to the Object Repository, if any are used in the project.
- `.screenshots` - Contains informative screenshots generated in UI automation activities, if any are used in the project.
- `.settings` - Contains activity project settings used at runtime.
- `.tmh` - Contains data related to test cases, if any are used in the project.

 **NOTE:**

Projects created with newer versions of Studio might not work with older Studio versions. Read more about [Backward and Forward Compatibility](#).

You cannot use the following characters in file names: ", <, >, |, :, \*, ?, \, /, ;. These characters come from [Microsoft Windows restrictions](#) and from [other special character restrictions](#).

## Setting the Project Language

Projects created in the Studio and StudioX profiles use the VB.NET language for expressions.

In the Studio Pro profile, when you create a new project, you can choose either VB or C#. VB is the language selected by default, but you can set C# as the default language for new projects by going to **Home** (Studio Backstage View) > **Settings** > **Design**. Please note that C# projects can only be opened from the Studio Pro profile.

The use of both VB and C# expressions in the same project is not supported. Neither is the use of VB expressions in C# workflows and vice versa. When you copy and paste activities from other projects, invoke or import workflows, make sure they use the same language as your project.

You can install C# libraries as dependencies in VB projects and vice versa. However, default values defined for arguments in the library project using language-specific expressions cannot be accessed from the project where the library is installed.

## C# Limitations

- The current C# implementation is based on the C# compiler that uses C# version 5, therefore access to newer features like coalescing assignment, null-conditional operator, null-coalescing operator, string interpolation and others is limited.
- Projects containing expressions with increments are invalid.
- Expressions containing the `nameof()` operator are marked as invalid and are not allowed with the current implementation of C#.
- Expressions containing calls to methods with optional arguments must include values for the optional arguments.
- Design time and runtime performance of C# projects is lower when compared to VB.NET. When runtime performance is essential, we recommend using VB.NET instead of C#.

## Setting the Project Version

### Semantic Versioning

The semantic versioning scheme has the format **Major.Minor.Patch[-Suffix]**, where:

- **Major** is the major version.
- **Minor** is the minor version.
- **Patch** is the patch version.
- **-Suffix** (optional) is a hyphen followed by a series of dot separated identifiers immediately following the patch version. This denotes a prerelease version.

Identifiers must comprise only of ASCII alphanumeric characters and hyphen, and they must not be empty. Numeric identifiers must not include leading zeroes. In addition, build metadata may be denoted by appending a plus sign and a series of dot separated identifiers immediately following the patch or pre-release version, for example 1.0.0-alpha+1.

When creating a new process or library, the default version scheme is semantic. It can be changed from the **Publish** window by simply adding an extra digit to the version number. The project's semantic version can be modified from the `project.json` file too. The patch

number 0 is automatically added to projects with version number `major.minor`.

## Legacy Versioning

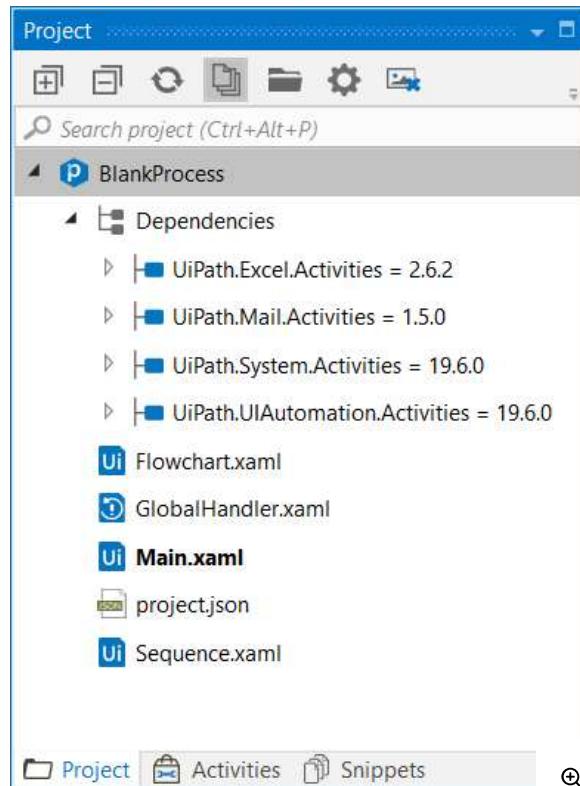
The legacy version number generated for the project has the format **M.m.bbbb.rrrr**, where:

- **M** is the major version.
- **m** is the minor version.
- **bbbb** is the build version.
- **rrrr** is the revision version.

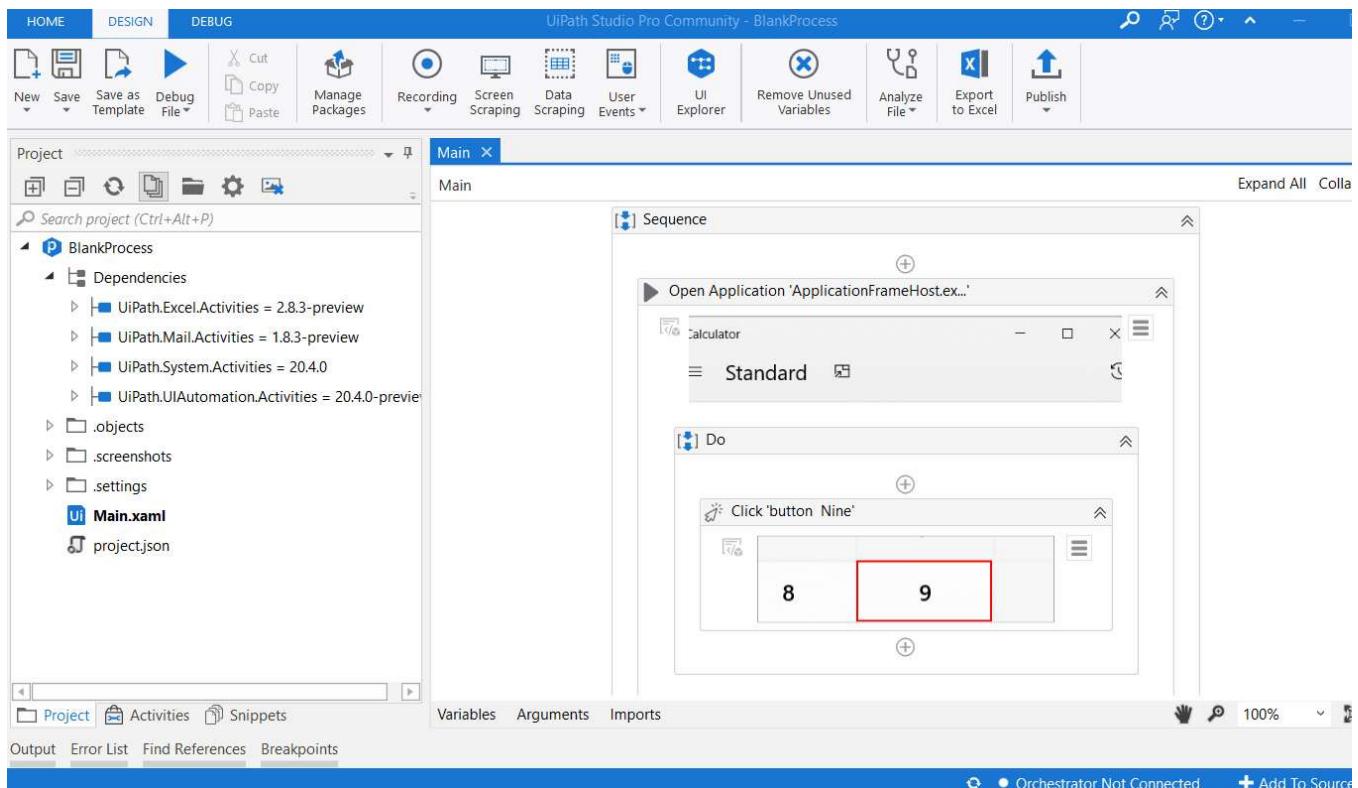
The major and minor versions can be edited in the `project.json` file also, while the build and revision versions are generated according to an algorithm - the build value is the number of days that elapsed since 01.01.2000. The revision value is the number of seconds which elapsed since the current day, until the moment of the release. The result is divided by 2, so that the maximum revision number **does not** exceed 65535.

The suggested version number in the **Publish** window is generated based on the project's previous versioning scheme, while the current date and timestamp are taken into account for projects using the 4-digit versioning scheme.

## Managing Projects



The **Project** panel enables you to view the contents of the current project, add folders, open the file location, manage dependencies, and adjust project settings.



Copy and paste files from File Explorer directly into the **Project** panel. The same can be done using drag and drop on one or multiple files, including .xaml workflows. Using **Ctrl + C** and **Ctrl + V** shortcuts you can also copy a file and duplicate it anywhere in the tree.

Option	Description
<b>Expand All</b>	Expands all nodes in the automation project.
<b>Collapse All</b>	Collapses all nodes in the automation project.
<b>Refresh</b>	Refreshes the project.
<b>Show All Files</b>	Shows all files belonging to the automation project, including the project.json.
<b>File Explorer</b>	Opens the project's location on the machine.
<b>Project Settings</b>	Opens the <b>Project Settings</b> window for processes or libraries.
<b>Remove Unused Screenshots</b>	Removes screenshots that aren't used when running the automation project.

## Context Menu for Projects

Right-click anywhere in the **Project** panel to open the context menu with options described in the following table. A different subset of options is available depending on where in the panel you right-click, the type of project, and whether the project is added to source control.

Option	Description
<b>Open Project Folder</b>	Opens the local folder containing the project.

Option	Description
<b>Project Settings</b>	Opens the <b>Project Settings</b> window for adjusting project preferences.
<b>Add</b>	Opens a list of items that can be added to the project: folder, <a href="#">sequence</a> , <a href="#">flowchart</a> , <a href="#">state machine</a> or <a href="#">global handler</a> .
<b>Import Workflows</b>	Imports .xaml files to the project and adds <b>Imported</b> in the file name if it coincides with the name of the main file.
<b>Import Files</b>	Opens the File Explorer window for importing various files into your project.
<b>Add to Source Control</b>	Adds the current project to source control using <a href="#">Git Init</a> , <a href="#">Copy to Git</a> , <a href="#">Add to TFS</a> , or <a href="#">Add to SVN</a> options. Please note that this option is only visible when right-clicking the project node. When a project is added to source control, additional options are available in the context menu. See the <a href="#">options for GIT</a> and the <a href="#">options for SVN and TFS</a> .
<b>Open</b>	Opens the selected files using the default program.
<b>Open File Location</b>	Opens the local folder containing the file.
<b>Rename</b>	Enables you to rename the selected file or folder, and opens the <b>Rename Item</b> window. The item is renamed in all occurrences.
<b>Delete</b>	Deletes the selected item only from your local machine.
<b>Select for Compare</b>	Selects the current file for comparison.
<b>Compare with Selected</b>	Compares the current file with the previously selected file using <a href="#">Compare Files</a> .
<b>Find References</b>	Finds all references to the file in the project. The results are displayed in the <a href="#">Find References</a> panel.
<b>Debug File</b>	Debugs the selected .xaml file.
<b>Set as Main</b>	Sets the selected .xaml file as <b>Main</b> in the project definition, meaning that the project execution starts with that file. There can only be one <b>Main</b> project file. The name of the file set as main appears in bold in the Project panel.
<b>Properties</b>	Open the library's <b>Properties</b> window for adding a tooltip and <a href="#">Help Link</a> .
<b>Set as Global Handler</b>	Sets the .xaml file as the <b>Global Exception Handler</b> for the project. This is applicable to one workflow per project/process.
<b>Remove Handler</b>	Removes the <b>Global Exception Handler</b> tag from the .xaml file.
<b>Enable Entry Point</b>	Marks the selected workflow file as an entry point for the process, making it possible to select it as the workflow to run first when using the <a href="#">Invoke Process</a> and <a href="#">Run Parallel Process</a> activities in other processes. <div style="margin-top: 20px;"> <b>NOTE:</b> <ul style="list-style-type: none"> <li>When a file is marked as an entry point, an arrow is displayed on the icon next to the file name .</li> <li>Enabling a file that is ignored from publishing as an entry point sets the file as publishable.</li> </ul> </div>

Option	Description
	<ul style="list-style-type: none"> <li>This option is not available for test case files.</li> <li>This option is not available in library projects.</li> </ul>
<b>Disable Entry Point</b>	No longer marks the selected workflow file as an entry point for the process. This option is not available for the workflow file that is set as Main.
<b>Ignore from Publish</b>	<p>Marks one or more selected files as excluded from publishing.</p> <p><b>NOTE:</b></p> <ul style="list-style-type: none"> <li>When a file is ignored from publishing, the icon next to the file name turns gray .</li> <li>Ignoring a file marked as an entry point from publishing disables the entry point.</li> <li>In a library project, a file that is ignored from publishing is included in the package, but no reusable component is created and made available in the <b>Activities</b> panel in projects where the library is installed as a dependency.</li> </ul>
<b>Set as Publishable</b>	Marks one or more selected files that are excluded from publishing as publishable.

## Context Menu Options for GIT

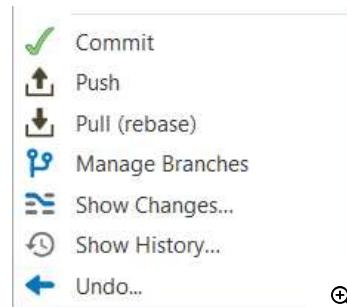
In projects added to GIT repositories, an icon is displayed next to each file in the **Project** panel to indicate the file status:

 The file is synced with the repository.

 The file has been modified.

 The file has been added.

Right-click a file or project node in the **Project** panel to open the GIT-specific context menu for [Managing Projects with GIT](#).



Option	Description
<b>Commit</b>	Commits current changes to the local GIT repository.
<b>Push</b>	Pushes the current version onto the remote repository.
<b>Pull (rebase)</b>	Pulls remote files and rebases the current branch.

Option	Description
<b>Manage Branches</b>	Opens the <b>GIT</b> window with options for managing currently added branches.
<b>Show Changes</b>	Opens the <b>File Diff</b> window for comparing changes between the local version and the remote version of the file.
<b>Show History</b>	Opens the <b>Show History</b> window for comparing two versions of the same file.
<b>Undo</b>	Opens the <b>Undo Pending Changes</b> window if the file was not committed or pushed to the remote repository.

## Context Menu Options for SVN and TFS

In projects added to SVN or TFS repositories, an icon is displayed next to each file in the **Project** panel to indicate the file status:

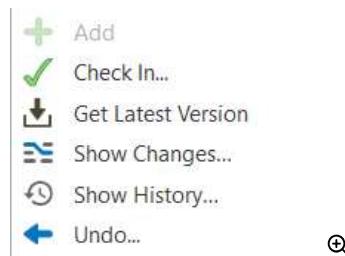
The file is not checked out for editing.

The file has been checked out for editing.

The file has been edited.

The file has been added.

Right-click a file or project node in the **Project** panel to open the context menu with options specific to managing projects with **TFS** or **SVN**.

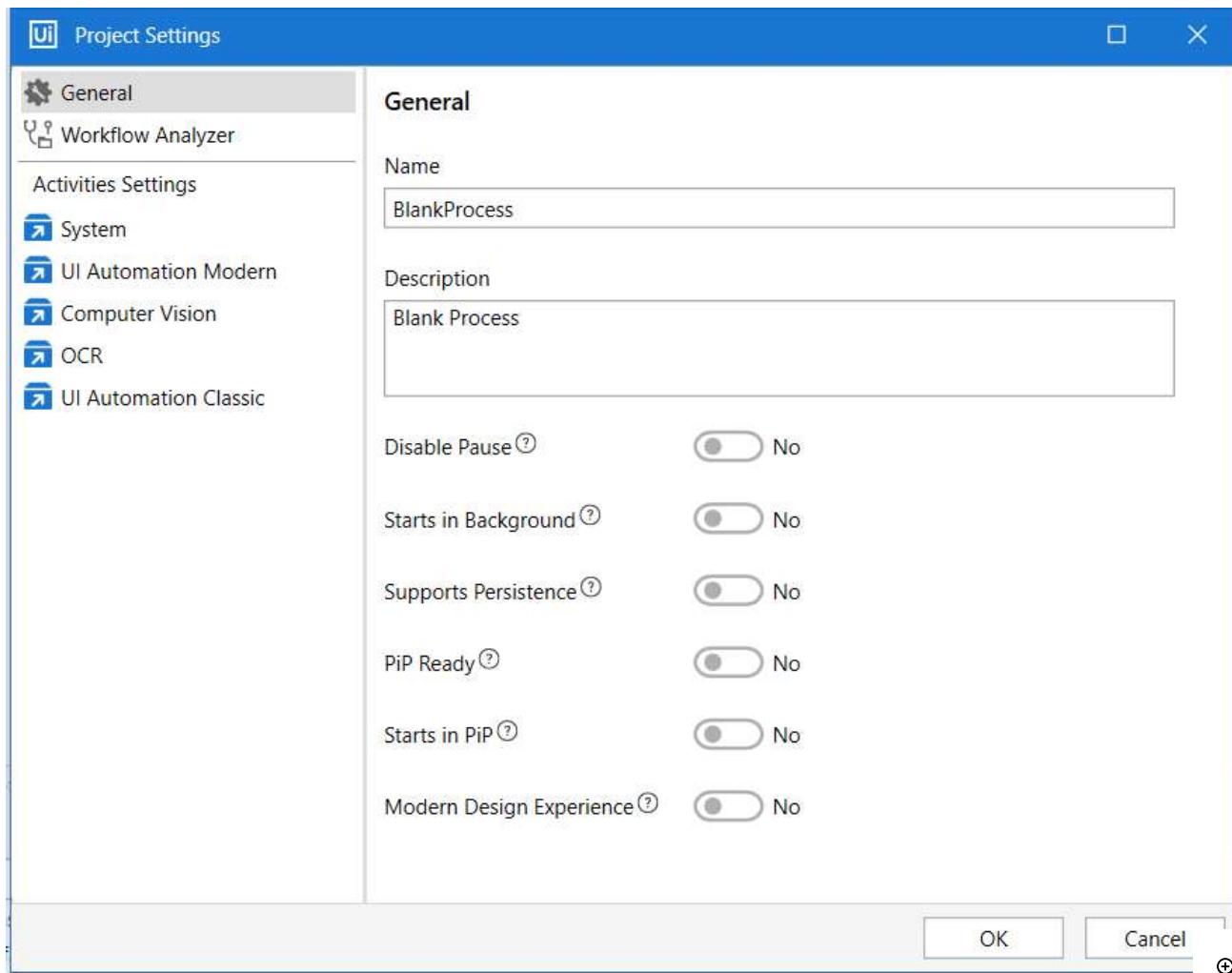


Option	Description
<b>Open</b>	Opens the selected .xaml file in the <b>Designer</b> panel, in read-only mode if it was not checked out for edit from the TFS/SVN repository.
<b>Rename</b>	Enables you to rename the selected file or folder, and opens the <b>Rename Item</b> window. When checking in the renamed .xaml file, the previously modified version must also be checked in.
<b>Delete</b>	Deletes the selected item only from your local machine. The latest checked in version of the file is still available in the TFS/SVN repository.
<b>Check Out For Edit</b>	Marks the selected file or folder as locked for editing. Checking out a file locks it on the server so that no one else can edit it.
<b>Finish Editing</b>	Checks in the project.json file in the repository, together with changes and a commit message.
<b>Add</b>	Uploads the selected item to the TFS/SVN server. This option is not available, if the item was previously uploaded to the server.
<b>Get Latest Version</b>	Downloads the latest version of the selected item from the TFS/SVN repository.

Option	Description
Show changes...	Opens the <b>File Diff</b> to compare changes between the versioned file and the one mapped locally.
Check In	Displays the <b>Check In Changes</b> window and enables you to upload the selected item to the server as the newest version. The .xaml file must be saved before uploading it. After it's checked in, the file becomes read-only in Studio.
Undo	Displays the <b>Undo Pending Changes</b> window and enables you to Revert the changes done to the project, either revert modified files to previous or unversioned states, or retrieve files which were deleted from the local machine. <b>Changes cannot be reverted after the file was checked in.</b>
Run	Runs the selected workflow, even if it's not checked out or added to the repository.
Set as Main	Sets the selected .xaml file as Main in the project. The first created .xaml is set as Main by default.

## Adjusting Project Settings

A set of individual settings can be established for each automation project that you're working on. Such settings are available in the **Project Settings** window, which can be opened by clicking the  in the **Project** panel.



## Field Description for the Settings Window

Field	Description
<b>Name</b>	Change the name of the project. Such names may contain whitespace characters. When naming projects, keep in mind that whitespace characters are removed at publish time. This field accepts up to 128 characters.
<b>Description</b>	Change the project description. This field accepts up to 500 characters.
<b>Disable Pause</b>	Enable or prevent users from pausing processes in the Robot tray. Set to <b>Yes</b> if pausing the process during execution would result in crashing it. For example, if an activity in your workflow uses the <b>Timeout</b> property, pausing the execution might cause the timeout to expire, thus breaking the execution.
<b>Starts in Background</b>	Set to <b>Yes</b> to turn the project in a <b>Background Process</b> and allow it to run in the background concurrently with other processes, as long as it does not use UI interaction.
<b>Supports Persistence</b>	Set to <b>Yes</b> to turn the project in an <b>Orchestration Process</b> .
<b>PiP Ready</b>	Set to <b>Yes</b> to indicate that the project was tested using <b>Picture in Picture</b> . If set to <b>No</b> , the option to use this feature is not available for the process.
<b>Starts in PiP</b>	Set to <b>Yes</b> to indicate that the process should be run by default using the Picture in Picture feature.
<b>Modern Design Experience</b>	Set to <b>Yes</b> to enable a <b>modern experience</b> of working with UI Automation, including new and improved activities, recorders, and wizards, as well as the <b>Object Repository</b> .

Click **OK** and the changes are made visible in the **Project** panel and `project.json` file.

Check out the [Configuring Activity Project Settings](#) page to read about how to adjust activity properties at project level.

**NOTE:**

Please take into consideration that whenever you wish to copy a large number of activities from one sequence to another, it is recommended to scroll down to the bottom of the Designer panel beforehand. This is due to a Windows Workflow Foundation limitation.

## About the Project.Json File

`Project.json` is an automatically generated file which is created for each `.xaml` file marked as **Main** in the project folder.

The file holds varied information, including project dependencies, or web services loaded in libraries. For more information about web services in libraries, check out the [Loading Web Services in Libraries](#) page.

**NOTE:**

**Manually editing the `project.json` file should be attempted for troubleshooting scenarios only, as it may lead to severe consequences and loss of support.**

As of 2018.2, you should specify a `project.json` file when running your project from the `UiRobot.exe` (command line) client.

The parameters contained in the `Project.json` file are described in the following table.

Parameter	Description
<code>name</code>	The title of the automation project. It is provided in Studio when creating a new process or library.
<code>description</code>	The description of the project. It is provided in the Description field in Studio when a new project is created.

Parameter	Description
main	<p>The entry point of the automation project. It consists of an .xaml file. The default name is "Main.xaml". It is displayed both as the title of the <b>Designer</b> panel and in the <b>Properties</b> panel in Studio. If you want to execute a different project first, change the value of this parameter to the name of the .xaml file to be processed.</p> <p><b>NOTE:</b> If your automation project contains multiple files, each of them should be linked to the Main.xaml file through the <a href="#">Invoke Workflow File</a> activity. This is especially useful when the project is published to Orchestrator and sent to a Robot, as the Robot executes only the file provided in this parameter.</p>
dependencies	<p>The activities packages used to create the automation project and their versions. The list is updated every time a dependency is added or removed from the project, or when a package version changes.</p> <p><b>NOTE:</b> Version numbers are composed of the following parts, in order: major, minor, build, and revision. The build value is the number of days that elapsed since 01.01.2000. The revision value is the number of seconds which elapsed on the day of the release, starting from 5 AM, GMT.</p>
webServices	<ul style="list-style-type: none"> <li>• namespace - The name of the service provided in the <b>Add new service</b> window for libraries.</li> <li>• serviceDocument - The path to the .xml or .json file containing metadata for the SOAP or Swagger service. The file is used when the service is repaired, and should be versioned as part of the project.</li> <li>• webDocumentUri - The file path or link to the Swagger or SOAP resource (provided when the service is created in the <b>Add new service</b> window).</li> <li>• uniqueReference - A reference needed for versioning the service.</li> </ul>
entitiesStores	<ul style="list-style-type: none"> <li>• serviceDocument - The path to the local entities file.</li> <li>• tenantName - The tenant where the Data Service resides.</li> <li>• namespace - The namespace under which the entities are imported (the project namespace).</li> <li>• uniqueReference - A reference needed for versioning the service.</li> </ul>
schemaVersion	The version of the project.json file.
studioVersion	The version of Studio used to create the automation project.
projectVersion	The version used when publishing this project to a feed. Represents the one set in the <b>Publish</b> window.
runtimeOptions	<ul style="list-style-type: none"> <li>• isPausable - Whether pause is enabled for the process</li> <li>• requiresUserInteraction - Whether the process is a long running process.</li> <li>• supportsPersistence - Whether the process supports persistence,</li> <li>• excludedLoggedData - Contains keywords that can be added to the name of an activity to prevent variable and argument values from being logged at the Verbose level. That can also be achieved by selecting the Private checkbox of any activity. Read more about the protection of sensitive information <a href="#">here</a>.</li> <li>• readyForPiP - Whether the process is marked as PiP ready.</li> <li>• startsInPiP - Whether the process is configured to start in PiP.</li> </ul>
designOptions	<ul style="list-style-type: none"> <li>• projectProfile - The Studio profile used when the project was created, either <b>Business</b> (StudioX) or <b>Development</b>.</li> </ul>

Parameter	Description
	<ul style="list-style-type: none"> <li>• <b>outputType</b> Shows the project type, either <b>Process</b>, <b>Library</b>, or <b>Tests</b>.</li> <li>• <b>libraryOptions</b> <ul style="list-style-type: none"> <li>• <b>includeOriginalXaml</b> - Whether to include original workflow files in the resulting .nupkg file.</li> <li>• <b>privateWorkflows</b> - The full name of private .xaml files contained in the library.</li> </ul> </li> <li>• <b>processOptions</b> <ul style="list-style-type: none"> <li>• <b>ignoredFiles</b> - List of RPA workflow files ignored from publishing.</li> </ul> </li> <li>• <b>fileInfoCollection</b> - Contains the following information for each test case file in the process: <b>editingStatus</b> (<b>Publishable</b> or <b>InProgress</b>), <b>testCaseId</b>, and <b>fileName</b>.</li> <li>• <b>modernBehavior</b> - Whether the process is configured to use the Modern Design Experience.</li> </ul>
arguments	<p>Contains the following information for each <b>input</b> and <b>output</b> argument defined in the workflow file that is set as Main: <b>name</b>, <b>type</b>, whether it is required, and whether it has a default value (<b>hasDefault</b>).</p> <p><b>NOTE:</b> Arguments information is added only to the file in the .nupkg package after publishing.</p>
expressionLanguage	The language set for the process (VisualBasic or CSharp).
entryPoints	<p>Contains the following information for each file marked as an entry point to the process <b>filePath</b>, <b>uniqueId</b>. and the following information for each <b>input</b> and <b>output</b> argument in the file: <b>name</b>, <b>type</b>, whether it is required, and whether it has a default value (<b>hasDefault</b>).</p> <p><b>NOTE:</b> Arguments information is added only to the file in the .nupkg package after publishing.</p>
isTemplate	Whether the project is a template.
templateProjectData	<p>Contains the following information for template projects:</p> <ul style="list-style-type: none"> <li>• <b>defaultProjectDescription</b> - The default project description for processes based on the template.</li> <li>• <b>defaultProjectName</b> - The default project name for processes based on the template.</li> </ul>

The following is an example of the contents of a project.json file included in a package published from Studio.

```
{
  "name": "UI-40028",
  "description": "Blank Process",
  "main": "Flowchart.xaml",
  "dependencies": {
    "UiPath.Excel.Activities": "[2.9.3]",
    "UiPath.Mail.Activities": "[1.9.3]",
    "UiPath.System.Activities": "[20.10.1]",
    "UiPath.UIAutomation.Activities": "[20.10.6]"
  },
  "webServices": [],
  "entitiesStores": [],
  "schemaVersion": "4.0",
  "studioVersion": "20.10.2.0",
  "projectVersion": "1.0.1",
  "uiAutomationVersion": "20.10.2.0"
}
```

```
"runtimeOptions": {
    "autoDispose": false,
    "isPausable": true,
    "requiresUserInteraction": true,
    "supportsPersistence": false,
    "excludedLoggedData": [
        "Private:*",
        "<em>password</em>"
    ],
    "executionType": "Workflow",
    "readyForPiP": false,
    "startsInPiP": false
},
"designOptions": {
    "projectProfile": "Developement",
    "outputType": "Process",
    "libraryOptions": {
        "includeOriginalXaml": false,
        "privateWorkflows": []
    },
    "processOptions": {
        "ignoredFiles": []
    },
    "fileInfoCollection": [],
    "modernBehavior": false
},
"arguments": {
    "input": [
        {
            "name": "argument1",
            "type": "System.String, mscorlib, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089",
            "required": false,
            "hasDefault": false
        },
        {
            "name": "argument2",
            "type": "System.String, mscorlib, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089",
            "required": false,
            "hasDefault": false
        }
    ],
    "output": []
},
"expressionLanguage": "VisualBasic",
"entryPoints": [
    {
        "filePath": "Main.xaml",
        "uniqueId": "5289efb0-f8bc-42f3-8cf4-0caa3a7d1915",
        "input": [
            {
                "name": "argument1",
                "type": "System.String, mscorlib, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089",
                "required": false,
                "hasDefault": false
            },
            {
                "name": "argument2",
                "type": "System.String, mscorlib, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089",
                "required": false,
                "hasDefault": false
            }
        ]
    }
]
```

```
        },
    ],
    "output": []
},
{
    "filePath": "Flowchart.xaml",
    "uniqueId": "d0904ba0-506e-437d-979c-b9da4325faad",
    "input": [
        {
            "name": "argument1",
            "type": "System.String, mscorelib, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089",
            "required": false,
            "hasDefault": false
        },
        {
            "name": "argument2",
            "type": "System.String, mscorelib, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089",
            "required": false,
            "hasDefault": false
        }
    ],
    "output": []
},
],
"isTemplate": false,
"templateProjectData": {},
"publishData": {}
}
```



Default Theme

English



# Studio User Guide

RELEASE:

2023.4



## TABLE OF CONTENTS

### [Input Methods](#)

## Input Methods

Input actions require you or the robot to directly interact with an opened application or web page. There are three types of input methods for click and type actions, that differ in terms of compatibility and capability.

We generally recommend the **Simulate Type/Click** method as it is the fastest of the three and works in the background, but only if you do not need to send special keyboard shortcuts. If this does not work for you, try the **SendWindowMessages** method and then the **Default** one, as it is the slowest.

Capability Method	Compatibility	Background Execution	Speed	Hotkey Support	Auto Empty Field
<b>Hardware Events</b>	100%	no	50%	yes	no

Capability Method	Compatibility	Background Execution	Speed	Hotkey Support	Auto Empty Field
<b>SendWindowMessages</b>	80%	yes	50%	yes	no
<b>Simulate Type/Click</b>	99% - web apps 60% - desktop apps	yes	100%	no	yes
<b>ChromiumAPI</b>	100% - Chrome, Edge browsers	yes	50%	yes	yes

 **NOTE:**

When the browser is started with **ChromiumAPI**, a ribbon shows up stating that the browser started in debug mode. This message does not show up if the extension is installed via [policy](#).

The input method can be changed at any point from the **Properties** panel of the selected activity. If the **SimulateType** or **SendWindowMessages** check boxes are not selected, then the **Default** method is applied.



Properties

UiPath.Core.Activities.TypeInto

**Common**

ContinueOnError	Specifies to continue executing the
DelayAfter	Delay time (in milliseconds) after t
DelayBefore	Delay time (in milliseconds) before
DisplayName	Type into

**Input**

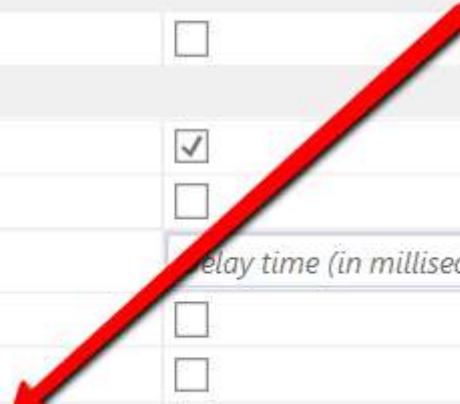
Target	Target
Text	"Seize the means of production!"

**Misc**

Private	<input type="checkbox"/>
---------	--------------------------

**Options**

Activate	<input checked="" type="checkbox"/>
ClickBeforeTyping	<input type="checkbox"/>
DelayBetweenKeys	Delay time (in milliseconds) betwe
EmptyField	<input type="checkbox"/>
SendWindowMessages	<input type="checkbox"/>
SimulateType	<input checked="" type="checkbox"/>



The **Default** application simulates a click or type with the help of the hardware driver, while the **Simulate Type/Click** method uses the technology of the target application. Lastly, the **SendWindowMessages** works by sending a specific message directly to the target application.



Default Theme

English



# UI Automation Activities

## TABLE OF CONTENTS

### [Click](#)

[Properties](#)

[Example of using the Click activity](#)

[Example of using the Double Click activity](#)

## Click

`UiPath.Core.Activities.Click`

Clicks a specified UI element.

### **IMPORTANT:**

The **Double Click** activity has the same functionality as the **Click** activity, the only difference is that for the **Double Click** activity, the **ClickType** is set by default on **CLICK\_DOUBLE**, while for the **Click** activity, the **ClickType** is set by default on **CLICK\_SINGLE**.

## Properties

## Common

- **ContinueOnError** - Specifies if the automation should continue even when the activity throws an error. This field only supports Boolean values (True, False). The default value is **False**. As a result, if the field is blank and an error is thrown, the execution of the project stops. If the value is set to **True**, the execution of the project continues regardless of any error.

 **NOTE:**

If this activity is included in **Try Catch** and the value of the **ContinueOnError** property is **True**, no error is caught when the project is executed.

- **DelayAfter** - Delay time (in milliseconds) after executing the activity. The default amount of time is 300 milliseconds.
- **DelayBefore** - Delay time (in milliseconds) before the activity begins performing any operations. The default amount of time is 200 milliseconds.
- **DisplayName** - The display name of the activity.

## Input

- **ClickType** - Specifies the type of mouse click (single, double, up, down) used when simulating the click event. By default, single click is selected.
- **MouseButton** - The mouse button (left, right, middle) used for the click action. By default, the left mouse button is selected.
- **Target.ClippingRegion** - Defines the clipping rectangle, in pixels, relative to the **UiElement**, in the following directions: left, top, right, bottom. It supports both positive and negative numbers.
- **Target.Element** - Use the **UiElement** variable returned by another activity. This property cannot be used alongside the **Selector** property. This field supports only **UiElement** variables.
- **Target.Selector** - Text property used to find a particular UI element when the activity is executed. It is actually a XML fragment specifying attributes of the GUI element you are looking for and of some of its parents.
- **Target.Timeout (milliseconds)** - Specifies the amount of time (in milliseconds) to wait for the activity to run before the **SelectorNotFoundException** error is thrown. The default value is 30000 milliseconds (30 seconds).
- **Target.WaitForReady** - Before performing the actions, wait for the target to become ready. By default, **Interactive/Complete** is selected. The following options are available:
  - **None** - Does not wait for anything except the target UI element to exist before executing the action. For example, you can use this option if you want to retrieve just text from a web page or click a particular button, without having to wait for all UI elements to load. Note that this may have unwanted consequences if the button relies on elements which are not yet loaded, such as scripts.
  - **Interactive** - Waits until only a part of the app is loaded.
  - **Complete** - Waits for the entire app to be loaded.

## Misc

- **Private** - If selected, the values of variables and arguments are no longer logged at Verbose level.

## Options

- **AlterIfDisabled** - If enabled, the action is executed even if the specified UI element is disabled.
- **CursorMotionType** - Specifies the type of motion performed by the mouse cursor. There are two available options:
  - **Instant** - The cursor jumps to the destination. By default, **Instant** is selected.
  - **Smooth** - The cursor moves in increments. Has no effect if **SendWindowMessages** or **SimulateClick** are enabled.
- **CursorPosition.OffsetX** - Horizontal displacement of the cursor position according to the option selected in the Position field.
- **CursorPosition.OffsetY** - Vertical displacement of the cursor position according to the option selected in the Position field.
- **CursorPosition.Position** - Describes the starting point of the cursor to which offsets from OffsetX and OffsetY properties are added. The following options are available: TopLeft, TopRight, BottomLeft, BottomRight and Center. The default option is Center.
- **KeyModifiers** - Enables you to add a key modifier. The following options are available: **None**, **Alt**, **Ctrl**, **Shift**, **Win**. The default option is **None**.

 **NOTE:**

**KeyModifiers** cannot be used with the **SimulateClick** or **SendWindowMessages** options. No error is thrown when executing a workflow that contains an activity with one of these combinations of options.

- **SendWindowMessages** - If selected, the click is executed by sending a specific message to the target application. This input method can work in the background, is compatible with most desktop apps, but it is not the fastest of the methods. By default, this check box is not selected. If neither this nor the **SimulateClick** check boxes are selected, the default method simulates the click by using the hardware driver. The default method is the slowest, it cannot work in the background, but it is compatible with all desktop apps.
- **SimulateClick** - If selected, it simulates the click by using the technology of the target application. This input method is the fastest of the three and works in the background. By default, this check box is not selected. If neither this nor the **SendWindowMessages** check boxes are selected, the default method performs the click using the hardware driver. The default method is the slowest, it cannot work in the background, but it is compatible with all desktop apps. If you select this property, it is recommended to check the state of the target UI element prior to execution. More details can be found on this [page](#).

 **NOTE:**

**SimulateClick** can only be used with the **CLICK\_SINGLE** and **BTN\_LEFT** input types.

Otherwise, an error is thrown when executing the workflow.

- **UnblockInput** - It should be used when **Click** triggers a modal dialog box or other blocking UI. Only works together with **Simulate Click**. This field supports only Boolean values. The default value for this option is **False**.

## Example of using the Click activity

[Here](#) you can see how the **Click** activity is used in an example that incorporates multiple activities.

## Example of using the Double Click activity

[Here](#) you can see how the **Double Click** activity is used in an example that incorporates multiple activities.



Default Theme

English