**Best Practice Guidelines for Development**

Practical Tips for UiPath Development

**Best Practice Guidelines for Development**

Practical Tips for UiPath Development

VERSION 1.01

AS OF NOVEMBER 16, 2020

**Table of Contents**

1. INTRODUCTION 3 1.1. Motivation 4

1.2. Positioning 4

1. PROGRAMMING GUIDELINES 4

2.1. Naming Conventions 5

2.2. Obsolete Statements 5

2.3. Layout 5

2.4 Workflow Analyzer 6

3. PERFORMANCE 6

3.1. Principle of Avoidance 6

3.2. Using Existing Tools 6

3.3. UI Activity Properties 7

4. ROBUSTNESS 8

4.1. Error Handling 8

4.2. Selectors 8

4.3. Logging 9

5. SECURITY AND COMPLIANCE 10

5.1. Audit Features 10

5.2. Security Defects 11

5.3. Compliance Standards 11

6. INFRASTRUCTURE AND LIFE CYCLE MANAGEMENT 12

6.1. Infrastructure 12

6.2. Change Management 13

6.3. Testing Robots 13

**1. INTRODUCTION**

UiPath Studio is an advanced tool that allows you create automation processes in a visual, drag-and-drop manner through diagrams. There is coding involved, but the beauty of UiPath is that it allows non-technical users to learn how to build great software robots, and it gives organizations the right governance tools to manage it all. You need a control flow for long processes. You need to know how to use if-statements and variable data types. And some issues can only be solved with invoking code or creating custom activities in either Visual Basic or C#, which obviously requires coding skills.

UiPath Robot executes the processes built in UiPath Studio, as a user would. Robots can work attended (run under human supervision) and unattended (run without human supervision in any environment, whether it is virtual or not).

UiPath Orchestrator is a web application that allows you to deploy, schedule, monitor and manage robots and processes, while business exception handling is available through centralized work queues. In Orchestrator, there are two machine entities that work as API key generators, which authorize the connection between UiPath Robot and Orchestrator. Orchestrator tracks and logs everything every robot does, along with everything people do with robots, so you can keep compliant and secure. The first thing you need to do in this web app is register your Robot(s). After that, you need to group them together in an environment. Workflows published to this platform have to be linked to an environment, and from there, you start executing: be it right now or based on a customly-defined schedule. You can run a process on one or multiple Robots, a specified number of them or on all the Robots that are grouped in an environment.

The Robotic Enterprise Framework (REFramework) is a project template based on state machines. It was created to fit all of the best practices regarding logging, exception handling, application initialization, etc. to tackle a complex business scenario. The template contains several state containers for initializing applications, retrieving input data, processing it and ending the transaction. All of these states are connected through multiple transactions, which cover almost every need in a standard automation scenario. There are also multiple invoked workflows, each handling certain aspects of the project. Consider using the REFramework if you are automating a process that fits this description of transactions, especially if you are using Orchestrator queues. It is not required to use the Orchestrator with the REFramework template, but you will need to change the input from being queues (from Orchestrator) to any other input.

**1.1 Motivation**

UiPath suggests scaling your automation program with a 7-pillar framework.

* Executive visioning
* Automation operating model
* Value creation and assurance
* Business outcome prioritization
* IT automation readiness
* Controls framework
* Citizen model and attended strategy

You can read about each pillar here: <https://www.uipath.com/blog/framework-scaling-automation-programs>.

**1.2 Positioning**

UiPath and specialized publishing houses already provide excellent publications on RPA development and functionalities in UiPath Studio. UiPath Academy has enrolled over 600,000 users around the world. The courses they offer are free, and they cover more than just development. Books like “Learning Robotic Process Automation: Create Software robots and automate business processes with the leading RPA tool – UiPath” and “Robotic Process Automation Projects: Build real-world RPA solutions using UiPath and Automation Anywhere” authors have already taken a huge step towards recommendations for utilizing RPA effectively.

The value of this document lies in the summary of best practices, practical tips, and proven system of rules from users. These guidelines can provide users, developers, development managers, project managers and IT managers with recommendations and assistance so that they do not need to re-invent the wheel. They can build on the experience gained by others. The recommendations provided by these guidelines do not claim to be complete nor to be generalized, but they do present a selection of practical tips.

**2. PROGRAMMING GUIDELINES**

This chapter describes recommended programming guidelines for robots that are created by means of the UiPath Studio IDE using drag-and-drop activities and VB.NET. This facilitates the maintenance of the code and/or enables an efficient corporation with different internal and external persons in the further development and maintenance of a robot.

**2.1 Naming Convention**

Name conventions describe the uniform and binding guidelines for the naming of software objects (e.g. workflows, sequences and activities in UiPath Studio’s case) and/or for naming the objects in the source code (e.g. variables). It is strongly recommended to specify a name convention that will be consistent across all platforms. Variables are consistently used in the Config file, Orchestrator and any systems like SAP that are used in the automated process. The purpose of using a standardized name convention is to considerably increase the maintainability of customer-specific adaptations and expansions. This leads to lower maintenance requirements and costs, and it results in a faster troubleshooting if there are any errors. To make any new employees familiar with the general rules and the specific requirements, the explicitly formulated name convention should be part of the internal training.

**BEST PRACTICE:** use variable names in camelCase so that they are easier to read.

**2.2 Obsolete Statements**

For the sake of organization and readability, it is recommended to remove any activities that are not contributing to the process. This includes removing any commented out activities that are not affecting the logic. It is also necessary to remove obsolete statements to reduce the size of the robot file and to improve the speed.

**BEST PRACTICE:** do not use any techniques for hiding your source code, especially in production.

**2.3 Layout**

When choosing how to design your robot from a high level, consider that a state machine is better at describing a continuously running mechanism. Flowcharts are better at describing a branching sequence of actions. State machines naturally encapsulate action groups, while flowcharts do not. Sequences are the smallest type of project. They are suitable to linear processes as they enable you to go from one activity to another seamlessly, and act as a single block activity. Flowcharts can be used in a variety of settings, from large jobs to small projects that you can reuse in other projects. The most important aspect of flowcharts is that, unlike sequences, they present multiple branching logical operators, that enable you to create complex business processes and connect activities in multiple ways. Avoid nested if statements whenever possible and use flowcharts instead for a tidier process.

**BEST PRACTICE:** establish the correct requirements to decide how to design the robot layout before starting development.

**2.4 Workflow Analyzer**

The 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. You can use this tool to flag warnings when there are inconsistencies in variable naming conventions, redundant sequences, nested if clauses, etc..

**BEST PRACTICE:** enable all rules in the Design Best Practices category of the Workflow Analyzer

**3. PERFORMANCE**

To avoid performance issues, the following rules should be taken into account in the course of UiPath Studio development. As is the case for other aspects of software development, performance will be improved if you know what should actually be done. If the reason for a piece of code is not obvious, this needs to be cleared up first.

**3.1 Principle of Avoidance**

“The most secure, fastest, most precise, cheapest, easiest to maintain, most reliable and easiest to document pieces of a computer system are those that aren’t there.” (Gordon Bell)

In order to follow this best practice, you should evaluate closely which code is really needed. Remember to use the Workflow Analyzer to assist with creating rules to warn you of any obsolete code.

**BEST PRACTICE:** avoid any unnecessary coding.

**3.2 Using Existing Tools**

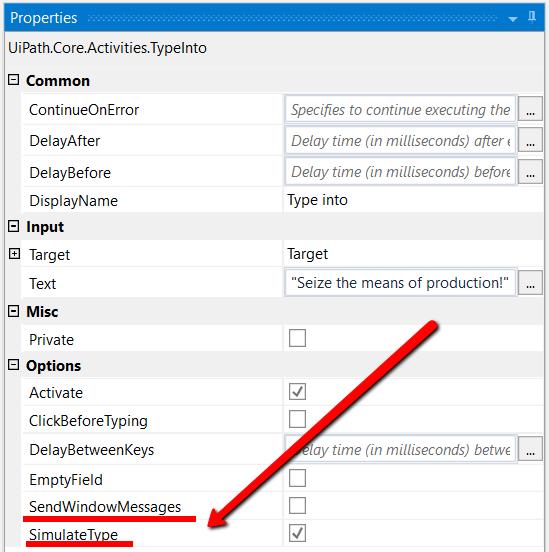
The tools available in UiPath Studio support the creation of performance-optimized robots as well as the analysis of performance issues. Orchestrator provides a monitoring solution which gives you real-time metrics to help you keep an eye on the health and state of your system. Be it in regard to robots, queues and queues SLA, or processes, you can check the state of your system in either the last hour or last day.

UiPath Insights is an RPA analytics solution that tracks, measures, and forecasts the performance of your entire automation program—so you can propel your automation journey to the next level. Insights utilizes columnar data storage instead of row-based storage, so it scales well with large amounts of data. With Insights, you have all of your operational data, plus any business data you’d like, to build any dashboards you want. All dashboards are exportable as PDFs, PNGs, and most widgets can be exported as a CSV file.

**BEST PRACTICE:** use the tools available to you to monitor and analyze how your robots perform

**3.3 UI Activity 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. Although these properties also contribute to robustness, they are necessary for performance. The input methods are especially important for managing the speed of certain activities. It is generally recommended to use 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.



**HINT:** If the ContinueOnError field of an activity inside a Try Catch is set to True, no error is caught when the project is executed.

**BEST PRACTICE:** utilize the UI activity properties to improve performance and robustness

**4. ROBUSTNESS**

In this chapter, we describe the measures developers need to take into account to create robust robots. Before we proceed, we must define the term robustness in the context of UiPath robots. The robustness of a program is the ability to run and produce correct results even under unfavorable circumstance. In particular, a robust program needs to recognize and handle errors in order to preserve the desired functionality.

**4.1 Error Handling**

UiPath Studio offers debugging functions similar to many IDEs. You can utilize steps, validation, breakpoints, etc. to identify and remove errors from the project. Exception handling is the more unique situation when using UiPath because Try Catch activities are the recommended way to handle errors. There are also rethrow, throw, and terminate workflow activities, which can be useful, but most of the time you can account for any potential errors with Try Catch activities. You can also use the Retry Scope activity for more fragile sections in a workflow, which retries the contained activities as long as the condition is not met, or an error is thrown.

**BEST PRACTICE:** implement sequences within the Try / Catch blocks that allow you to restore the workflow. Include an activity like Send Outlook Mail Message to email any errors with the location and description of the error.

**HINT:** if using a retry scope activity, only the Element Exists activity is allowed in the Condition section when exiting a loop.

**4.2 Selectors**

In the case of robustness, selectors are important to perfect because a bad selector will cause issues when the slightest change occurs. When you develop a website, you have the same freedom that someone developing an RPA process has. You have multiple ways to get to the same place, and you have the freedom to name your different variables as you please, but a good selector digs deep into the descriptors or directions. You are giving your robot step-by-step directions on what to find and click in that confusing code that makes up that site, including what you want to click.

UIExplorer is the best way to tune your selectors. It 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

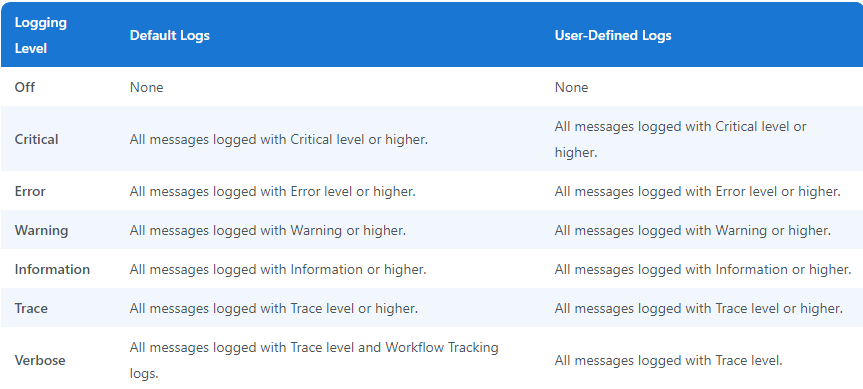
UiPath.UIAutomation.Activities package is installed as a dependency to the project.

**BEST PRACTICE:** use UiExplorer to understand the visual tree in your selector and identify where you can improve certain properties to find your element on the screen.

**HINT:** instead of clicks, try to use keyboard shortcuts whenever possible.

**4.3 Logging**

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. Messages are logged on the following levels:



**HINT:** the priority order of the log types is: Verbose<Trace<Information<Warning<Error<Critical<Off.

Default logs are 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. The events logged by this category are:

* **Execution Start** – every time a process starts (Information level)
* **Execution End** – every time a process ends (Information level)
* **Transaction Start** – every time a transaction within a process starts (Information level)
* **Transaction End** – every time a transaction within a process ends (Information level)
* **Error Log** – every time the execution encounters an error and stops (Error level)
* **Debugging Log** – if the robot logging setting is set to Verbose and contains activity names, types, variable values, arguments, etc. (Trace level)

User-defined logs are generated according to the process designed by the user in Studio, when using the Log Message activity or the Write Line activity. These logs are in the form similar to a JSON key-value pair (“field1:value1,” “field2:value2”). They are sent to the UiPath Orchestrator, where a few extra fields are added. It is recommended to learn to use log fields with Orchestrator for analytic dashboards. UiPath provides out-of-the-box dashboards for PowerBI and Tableau on their marketplace (UiPath Go!).

**BEST PRACTICE:** use log severity levels to differentiate severe and important events from irregular or even regular events. Do not dismiss lower severity issues. They can be used as data points when trying to create a baseline for process behavior. Your log files should contain mostly Information and Warning messages.

**5. Security and Compliance**

This chapter describes how programming errors can have a negative effect on the security of a company. We talk about a security defect when a programming error causes an unwanted side effect, through which an unauthorized user can act maliciously. We talk about a compliance violation when a user can bypass an audit-relevant security mechanism due to a programming error. An important difference is that security defects in code give rise to attacks by hackers, while compliance violations in code potentially violate legal requirements and/or cause red lights in financial reporting or audits.

**5.1 Audit Features**

The Audit page displays the audit trail for actions performed by all entities in Orchestrator. The data in the Audit page can be filtered by several criteria, namely Component, User, Action, and Time. Component filters by components like assets, comments, credential stores, environments, etc.. The user filter enables viewing changes performed by a certain user. The action filter contains actions that the Audit page logs details about like Assign, Create, Delete, Import, etc.. The time filter makes it possible to view the events according to the time they happened. The available intervals are last hour, last day, last week, and last 30 days.

HINT: the entire information comprised in the Audit page can be downloaded as a .csv file by clicking the Export button.

5.2 Security Defects

Because UiPath development is still in its youth relative to other software with long histories of vulnerabilities, there are not many recorded instances of security defects. However, they are still important to recognize. UiPath Orchestrator before 2018.3.4 allowed CSV injection related to the Audit export, Robot log export, and Transaction log export features. UiPath was able to recognize this defect and fix the issue before anyone could abuse it. Another notable vulnerability was UiPath Orchestrator through 2018.2.4 allowed any authenticated user to change the information of arbitrary users (even administrators) leading to privilege escalation and remote code execution. The defect requires an attacker to be logged into the system (such as at a command line or via a desktop session or web interface). UiPath was able to resolve this issue as well. It is recommended to report any security issues or product abuse issues to UiPath’s Security and Product teams.

BEST PRACTICE: use the SecureString variable to store credentials. After a SecureString is retrieved, it is recommended to be logged into applications with the Type Secure Text activity, which sends a SecureString to a UI element. Use the Send Keys Secure activity for terminals. It is important for the credential to not be used for any purpose other than the intended one. Variables holding such secure strings should be defined at the narrowest scope possible. Arguments should ideally not be used for passing credentials from one workflow to another.

5.3 Compliance Standards

Path has created an internal code of conduct available here: <https://www.uipath.com/hubfs/legalspot/UiPath%20Global%20Code%20of%20Conduct.pdf>. The internal code of conduct is applicable to all employees, freelancers, those employed by carriers or other contingent workers acting on behalf of UiPath or having access to UiPath systems including its subsidiaries and affiliates to ensure they act in good faith. A global partner code of conduct is available here: <https://www.uipath.com/hubfs/legalspot/2020.Apr.24_UiPath_Business_Partner_Code.pdf>. This code of conduct sets the expectations and defines the minimum standards of business conduct and business practices applicable to all UiPath clients, resellers, consulting partners, vendors, OEMs, suppliers, agents, entities and/or individuals who do business with or on behalf of UiPath (the “partners”). All partners are expected to comply with this code of conduct.

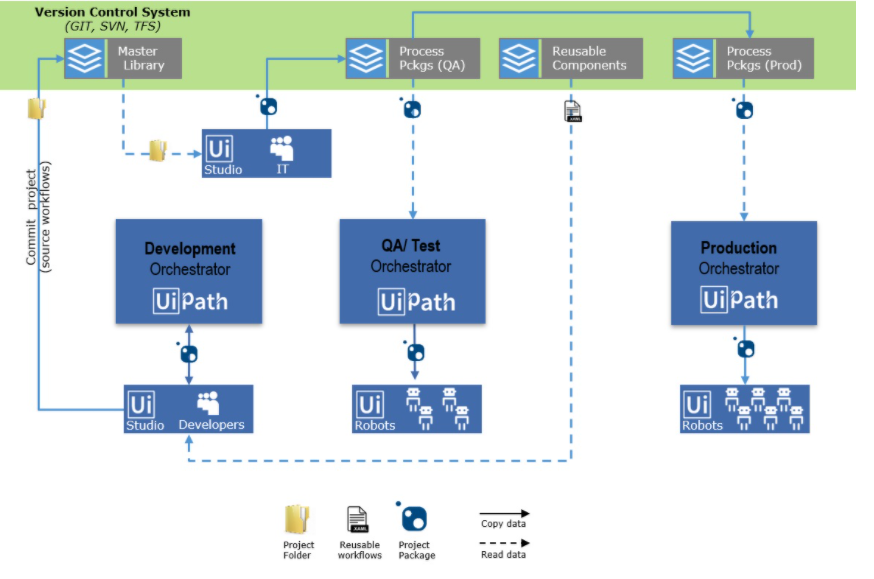
BEST PRACTICE: review your UiPath development to ensure it falls in line with UiPath standards, especially if your work will be audited at any point.

6. Infrastructure and Life Cycle Management

This chapter discusses the necessary infrastructure and life cycle management when implementing UiPath for an organization. You can find hardware and software requirements here: <https://docs.uipath.com/installation-and-upgrade/docs/robot-hardware-and-software-requirements>.

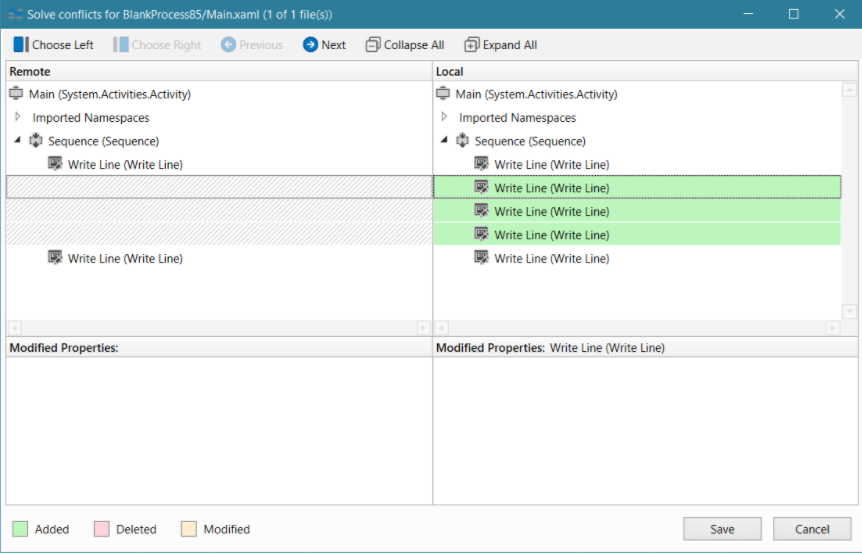
6.1 Infrastructure

A UiPath landscape usually consists of three separate systems. Similar to many software systems, we recommend a development system, quality assurance system and production system. It is of course possible to only implement a development system and production system and to handle testing in the development system, but it can be difficult to handle testing in the development system while developers are working on development. If you have a quality assurance testing system, you can test without interfering with developers’ time for development. Absolutely no development should be done in the quality and production systems.

BEST PRACTICE: use a network drive for your development in case hardware fails, and remember to always backup your development.

6.2 Change Management

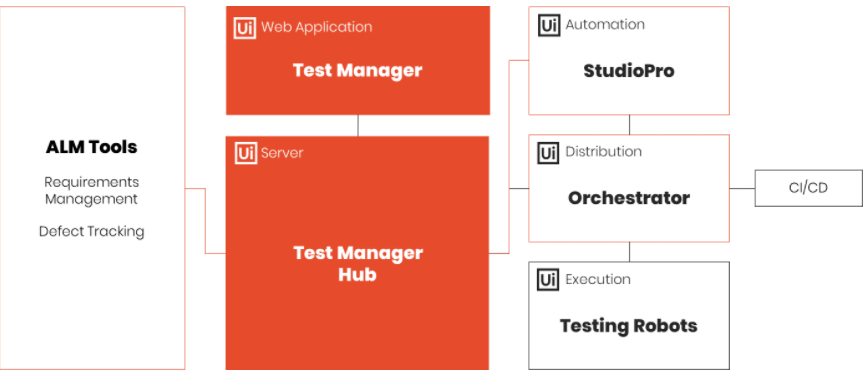
In order to keep a UiPath landscape maintainable and controllable, a formal change management process is recommended. Make sure to document all change requests including who requested the change, the amount of time allowed to work on the change, and all requirements for the change. For now, you can save your development in a network drive to automatically save changes in the production PC. Regarding deployment, it is possible to just manually copy and paste projects with changes to the quality and production environments, and UiPath is currently considering ways to automate this deployment. Regrading version control, we recommend using GIT integration in UiPath Studio to clone your repository and using commit and push functions to make changes to your development. You can create and manage branches just like you would in other languages. GIT also provides nice tool to solve conflicts, similar to the Workflow Diff tool in UiPath.



BEST PRACTICE: use GIT for version control, and use a network drive for deploying changes to production. Make sure to document all changes with a formal change management process.

6.3 Testing Robots

In order to ensure the testability of robots, the test requirements need to be determined early in the development process. To enable the testability of existing code often requires a large amount of effort. Furthermore, enabling testability does not result in new functionality and is therefore seen to be a low priority. Depending on the complexity of your robot, you might be able to just use test data and run your robot in Studio while taking screenshots of successful testing. But for more complex robots, you might want to automate your testing. In most cases, it’s recommended to use the UiPath Test Suite with Studio Pro to create automated tests. You can also schedule tests with the Orchestrator.



BEST PRACTICE: use the UiPath Test Suite with Studio Pro to automate your testing.