**UIPATH RPA CoE**

**UIPATH RPA Best Practices Guide**

This document is to understand the best practices to be followed during design and the development of the bots

**Artifact Contributors**

|  |  |  |
| --- | --- | --- |
| Name | Date | Author |
| Sukannya Jamadar | 2/25/2019 | Initial Document Author |
|  |  |  |
|  |  |  |
|  |  |  |

**Document Version History**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Change Description |
| 0.1 | 2/25/2019 | Sukannya Jamadar | Original Draft |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[1. Introduction 4](#_Toc2078452)

[2. Project Structure 4](#_Toc2078453)

[3. Folder Structure(RE-Framework) 4](#_Toc2078454)

[4. Naming conventions 5](#_Toc2078455)

[5. Commenting and Annotations 5](#_Toc2078456)

[6. Logging 5](#_Toc2078457)

[7. Exception Handling 6](#_Toc2078458)

[8. Decision 6](#_Toc2078459)

[9. Background Automation 7](#_Toc2078460)

[9.1 Input/Output Methods 7](#_Toc2078461)

[9.2 Selectors 7](#_Toc2078462)

[9.3 Containers 8](#_Toc2078463)

[9.4 Delay/Wait 8](#_Toc2078464)

1. **Introduction**

The purpose of this document is to understand the best practices to be followed during development of UiPath projects.

1. **Project Structure**

* UiPath project folder consists of subfolders (e.g. local and screenshots) and files (.xaml & .json).
* The workflows created have .xaml file extensions.
* Pick appropriate layout for your workflows:
  + Main -> flowchart/State machine
  + Business logic -> flowchart
  + UI interactions, navigational flows and Data processing -> Sequences
  + Avoid Nested IF statements, use flowchart instead.
* Break the whole process into smaller workflows.
* Build reusable workflows and call them into ‘Main’ workflow using *Invoke* activity.
* Keep environment settings in config file (.xlsx or .xml or .json) or in Orchestrator assets if they may change often.
* Do not store credentials in workflow directly, instead load them from local Windows Credential Store or Central Orchestrator.

1. **Folder Structure(RE-Framework)**

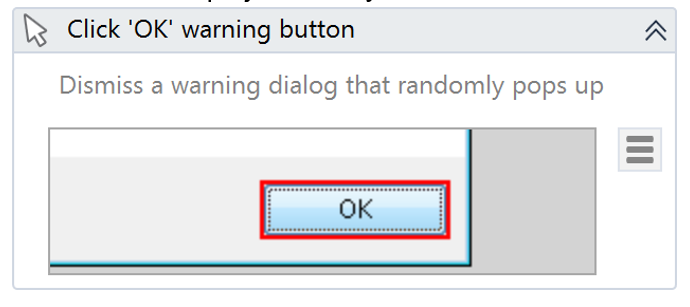
* Project
  + Main.xaml
  + Framework
    - <Common workflow by RE Framework>
  + Subfolder for Business Common Module
    - <Business Common workflow >
  + Sub Folder for Business Defined Category
    - <Workflow>
  + Test Framework

1. **Naming conventions**

* Always use descriptive and accurate names.
* Avoid using global variables and pass arguments instead.
* Workflow names should contain the verb describing the task it performs, e.g. GetTransactionData
* Activity names should describe the action taken, e.g. “Click id12” should instead be called “Click Save Button”
* Use camelCase (‘intEmployeeId’) for variables and TitleCase (GetData) for arguments and Workflows.
* Avoid single character variable names. Never use “i” or “x” for example.
* Do not use numbers, underscores or hyphens in variable names.
* Avoid using GenericValue variables unless is it necessary.
* Avoid magic numbers and hard-coding. Use environment variables, enums or constants as per context instead.
* Variables should have a data-type prefix, in lower case:
  + int: ‘integer’
  + str: ‘string’
  + b: ‘boolean’
  + lst: ‘list’
  + arr: ‘array’
  + dt: ‘datatable’
* Arguments should have direction (in/out) as prefix, starting with lower case
  + in: ‘In argument’
  + out: ‘Out argument’
  + io: ‘InOut argument’

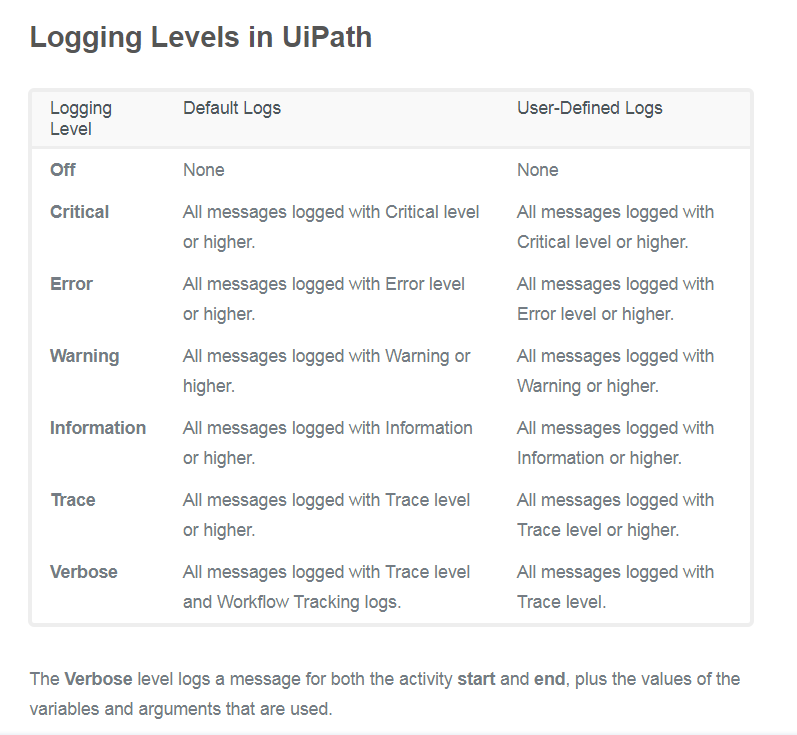
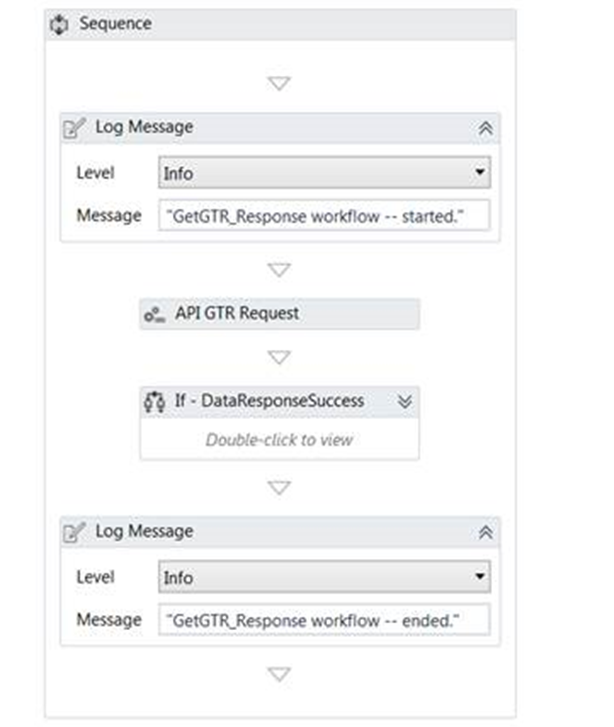
1. **Commenting and Annotations**

* Use comment activities to add comments into the workflow. Prefer logging over comments if possible.
* Annotations should be used to describe in more detail a technique or particularities of a certain interaction or application behavior.



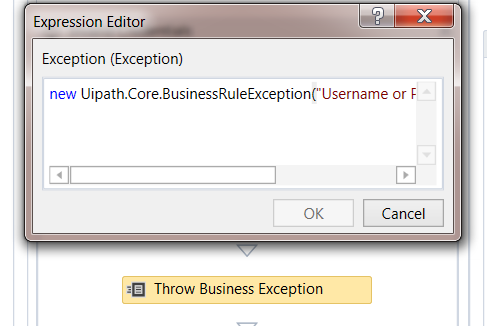
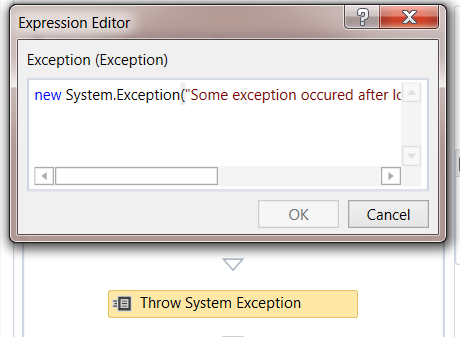
1. **Logging**

* By default, *Execution* and *Studio* logs are stored locally in ‘%LocalAppData%/UiPath/Logs’
* Log Message and WriteLine activities should be used. WriteLine activity logs with Trace level.
* Logging should be used
  + at the beginning and the end of every workflow – with “Trace” Level.
  + when data is coming in from external sources – with “Info” Level.
  + each time an exception is caught at the high level – with “Fatal/Error” Level.
* Use MessageBox activity in development environment and not in production (for unattended bots).
* Logging level should be selected based on below conditions:

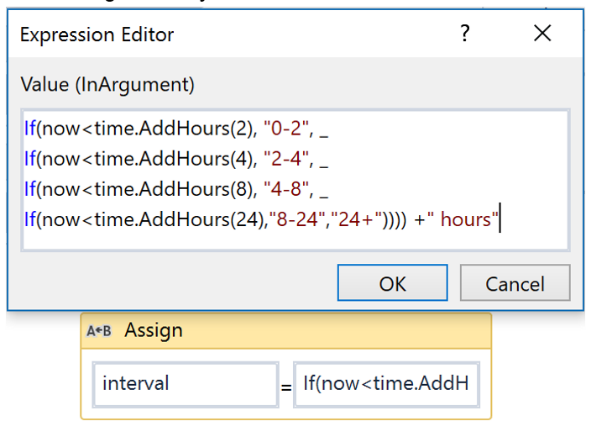
 

1. **Exception Handling**

* Always use Try/Catch for workflow or sequences where there are chances of errors being thrown.
* Use ***R****obotic* ***E****nterprise* ***Framework*** *Template as it proposes* flexible high-level overview of a repetitive process with Orchestrator integration, basic error handling and progress reports.
* Always use Try/Catch for workflow or sequences that have a probability of some exception.
* Prefer ***R****obotic* ***E****nterprise* ***Framework*** *Template* to designbigger workflows having extensive exception handling and provides readymade templates for Orchestrator integration, basic error handling, progress reports, etc.
* Put externally invoked workflows in Try Catch.
* Implement Recover Sequences in the event of an exception.
* Close all the applications, windows and webpages after the process execution is over.
* Throw exceptions based on type of exception (Application, Business, System).

1. **Decision**

* IF activity should be for short balanced branches.
* Decision activity is useful for nested IF ELSE conditions.
* IF Operator should be used for minor local conditions or data computing in expressions.
* Switch activity should be used instead of IF ELSE conditions. 

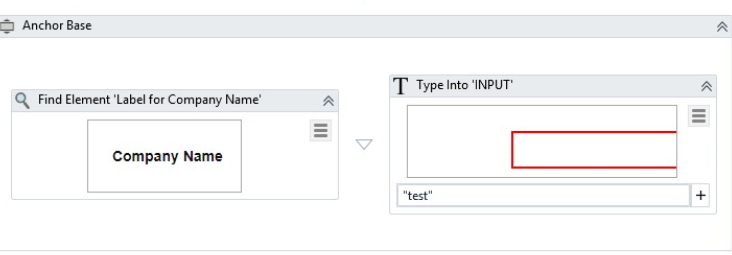
1. **Background Automation**

## 9.1 Input/Output Methods

* SimulateType or SimulateClick should be checked/enabled when we need UiPath to hook into the application and triggers the event handler of a specific UI element.
* SendWindowMessages should be checked/enabled when we need UiPath to post the event details to the application’s window which dispatches it to the target UI element internally.
* Use SimulateType, SimulateClick and SendWindowMessages options for navigation and data entry via Click and TypeInto activities
* Use SetText, Check and SelectItem activities for background data entry.
* GetText, GetFullText and WebScraping are output activities that run in the background.
* Avoid using ‘OCR’, if “OCR’ result is accurate then consider using TEXT activities.

## 9.2 Selectors

* Selectors should be able to identify the specific element irrespective of the environment.
* Selectors should be login independent.
* Wildcards (\*) must be used whenever selectors don’t have consistent attributes.
* Variables must be used in selectors when the input value might change as per business requirement.
* Selectors like “<html title=’\*’>” should not be used as this will identify multiple elements. Rather “<html title=’Google Calendar-\*’>” should be used as this identifies a specific element.
* Avoid using ‘idx’ in a selector and it should always be the last resort and even then, small numbers like 1 or 2 may be used.
* Use ‘Anchor Base’ instead of ‘idx’ attributes whenever possible.



## 9.3 Containers

* Developers should use containers wherever possible as this optimizes the code and makes selector identification easier.
* ‘Attach Browser’ activity should be used for UI automation using browsers having to perform multiple actions on the same browser.
* ‘Attach Window’ activity should be used for UI automation using desktop based applications.
* Add ‘Maximize’ activity in the containers.
* For multiple Browsers, we should use GetAttribute activity to get process ID of the browser and use that in selectors.

"<html pid='" + browserPid + "' />"

## 9.4 Delay/Wait

* Avoid using Delay activity, DelayBefore and DelayAfter.
* TimeOut field can be used instead of hardcoded delays.
* Use ElementExists to verify application state.
* Use ‘ElementExists’, ‘FindImage’, ‘FindText’, ‘OnImageAppear’, ‘OnElementAppear’, ‘WaitElementVanish’, ‘WaitImageVanish’, ‘WaitScreenText’ etc. along with properties like ‘WaitForReady’, ‘TimeoutMS’, ‘ContinueOnError’ etc. for UI Synchronization.