UiPath
Cheat Sheet

Layout Diagrams

Used to integrate **activities** into workflow design.

i. Sequence
- Linear representation of activities that follow each other in a fixed order.
- Easy to understand & is suited for simple & small scenarios.

ii. Flowchart
- Each step is represented by different symbols connected by arrows.
- Flexible & can showcase decision points, ideal for more complex workflows.

iii. State Machine
- Represented by flowcharts with conditional arrows called transitions (State Diagrams).
- Suited for high-level process diagrams of transactional business process templates.

Control Flow

The **order** in which particular actions are taken with the help of loops that help automate **repetitive tasks**.

i. If
- Contains a statement & two conditions.
- The Then section is executed if statement is True, Else section if it is False.

ii. While
- If the condition is met, actions in the body are executed.

iii. Do While
- Actions are first executed, followed by the condition. If the condition is met, the actions are performed again.

iv. For Each
- Iterates through a list of items, one at a time, and executing actions in the body of the loop.

Flow Decision

An activity that executes either one of the two branches, by default named True & False.

Execution depends upon whether the condition is met or not.

It is equivalent to the If activity but can only be used in Flowcharts.

Operators

Some common operators for various purposes in programming.

i. Assignment / Comparison
- **=** (equals)
- **> / >=** (greater than / greater than or equal to)
- **< / <=** (less than / less than or equal to)
- **<>** (not equals)

ii. Mathematical
- **+** (addition)
- **-** (subtraction)
- ***** (multiplication)
- **/** (division)

iii. Boolean
- **NOT**
- **AND** (&&)
- **OR** (||)

Variables

Store data and pass them between **activities**.

Can be created from the **Context Menu** with keyboard shortcut (CTRL + K), an **Assign activity**, or from the **Variables Panel.**

i. String
- Text of any kind ("aBc123@#$")
- Must be placed within quotation marks (" ").

ii. Int32
- Whole numbers (1, 55, 999)
- Storage capacity of 32 bits.

iii. Boolean
- True or False

iv. DateTime
- Dates & Times ("yyyy/MM/dd" – format can be changed).
- Use the .Now function for the actual date & time.

iv. Generic
- Any type of data (text, numbers, datetimes)
- Advantage:
  1. Convenience, flexible use of variables.
  2. No type considerations.
- Disadvantage:
  1. Lack of specific handling methods. (String manipulation methods cannot be used directly as they only work for String variables)
  2. Imprecise expression evaluations.

Datatables

A type of variable that can store big pieces of information, and act as a database or spreadsheet with **rows** and **columns**.

Commonly used in extraction of structured data from websites, or Excel files.

i. Initializing a datatable
- dt_1 = New System.Data.DataTable

ii. Filtering a datatable
- **Select** method can be used, returns an array of Datarows.
- dt_Array = dt_1.Select("Age='30'")

Arrays

A collection that can store multiple values of one of the many data types, with a **fixed** size.

i. Initializing an array
- strArray = new System. String () {}
- where () is the length & {} contains the values in the array.
- String [ ]

Lists

Similar to arrays, but, with a **flexible** size, making it more versatile.

i. Initializing a list
- strList = new System.Collections.Generic. List (of String)
- List <String>
- Items can be added using an Add To Collection activity.

Arguments

A kind of variable that also stores data but passes them between **workflows / projects** instead of just between **activities**.

Can be created in the **Arguments Panel**.

Mandatory fields when creating arguments:
Name: Denomination of the argument.
Direction: Direction of the argument.
Argument Type: Data type it stores.

i. In
- Can only be used within the given workflow.

ii. Out
- Can be used to pass data outside the given workflow.

iii. In / Out
- Can be used both within and outside the workflow.

Data Manipulation

Usage of some common predefined methods for Strings and others.

Let **str** be a string variable with value: "Hello World! "

i. Trim
- str.Trim()
- Removes leading & trailing spaces.
- *Result: "Hello World!"*

ii. Split
- strA.Split({" "},StringSplitOptions.None)
- Splits the string by a spacing and store each part into a string array.
- *Result: strA(0) = "Hello", str(1) = "World!"*

iii. Substring
- str.Substring(0,5)
- Takes a substring of the string starting from index 0 with a length of 5.
- *Result: "Hello"*

iv. Remove
- str.Remove(0,5)
- Takes a substring to remove instead of keep, starting from index 0 with a length of 5.
- *Result: "World"*

v. Replace
- str.Replace("!","~")
- Replaces '!' found in the string with '~'
- *Result: "Hello World~ "*

vi. Contains
- boolVar = Str.Contains("o")
- Checks whether the string contains the letter "o" and returns a Boolean value based on the result.
- *Result: boolVar = "True"*

vi. ToString
- intAge.ToString()
- Converts the variable type to a string.

vii. CInt
- Cint(str)
- Converts the variable type to an integer.

viii. Environment.NewLine
- "Line1: " + str + Environment.NewLine + "Line2: Hey!" >
- Generates a line break Content afterwards will be on the next line.
- *Result: Line1: Hello World!*
  Line2: Hey!

Selectors

Store attributes of a graphical user interface element and its parents.

Can be created automatically by using the **Attach to Live Element** feature or manually from **UiPath Explorer**.

i. Full Selectors
- Contains all the elements needed to identity an UI Element, including the top-level window.
- Recommended when switching between multiple windows.

ii. Partial Selectors
- Does not contain information about the top-level window.
- Activities containing partial selectors are enclosed in a container that contains a full selector of the top-level window.
- Recommended when performing multiple actions in the same window.

Wildcards

Symbols that allow dynamically-changing attributes in a selector by replacing character(s).

i. Asterisk (*)
- Replaces zero or more characters.

ii. Question Mark (?)
- Replaces a single character.

Recordings

Record and replay actions for automation, with the ability to modify & parametrize the recorded sequence.

Certain activities cannot be recorded such as **Keyboard shortcuts**, **Mouse hovers**, and, **Right-Clicks.**

F2 can be used to pause the recording for 3 seconds.

i. Basic
- Generates full selectors for each activity without a container, resulted workflow is slower than those with containers.
- Suitable for single activities.

ii. Desktop
- Generates a container with the selector of the top-level window and, partial selectors for each activity.
- Suitable for all types of desktop apps and multiple actions.

iii. Web
- Designed for recording in web apps & browsers & generates containers.
- Simulate Click/Type input methods by default.

iv. Citrix
- Designed for virtualized environments or SAP, permits only image, text & keyboard automation.
- Requires explicit positioning.

Excel Automation

Some of the activities that are used in Excel automation.

UiPath.Excel.Activities package required.

Excel activities in the scope require Excel to be installed & opened.
Workbook activities does not. (Works in the background)

i. Excel Application Scope
- Container that enables you to work with other Excel activities & where you specify the .xlsx file to work with.

ii. Read Range / Cell
- Reads the specified Excel file /Cell and stores it to a DataTable / String variable.

iii. Write Range
- Writes data from a DataTable to an existing Excel file, creates a new one if it does not exist.
- Overwrites existing data.

iv. Append Range
- Appends data from a DataTable to an existing Excel file, creates a new one if it does not exist.
- Does not overwrite existing data.

v. Insert / Delete Column
- Insert or delete a column from an Excel file or DataTable having specified the Column Name & Sheet Name.

vi. Output Data Table
- Writes a DataTable into a String using CSV format.

PDF Automation

Some of the activities that are used in PDF automation.

UiPath.PDF.Activities package required.

i. Read PDF Text
- Reads all characters from a specified PDF file & stores it in a String variable.
- Preferred activity as Read PDF With OCR is error prone.

ii. Read PDF With OCR
- Reads all characters from a specified PDF file using OCR technology & stores it in a String variable.
- Use only if required to extract text in an image of the PDF.

iii. Anchor Base
- When looking to extract specific values, use the Anchor Base activity.
- Works well with a Find Element / Image activity as the anchor (for handling structural changes), followed by a Get Text to extract the value.

Screen Scraping

Another method for extracting data from documents (e.g. PDF files) using the Screen Scraping Wizard.

i. FullText
- Default method, the fastest and the most accurate.
- Works only with desktop applications.

ii. Native
- Able to extract screen coordinates of the text.
- Works with applications that are built to render text with GDI.

iii. OCR
- Not as accurate but can extract text which the two other methods cannot.
- Has different OCR engines such as Google Tesseract & Microsoft Modi.

Email Automation

Some of the activities that are used in Email automation.

UiPath.Mail.Activities package required.

i. Save Mail Message
- Saves the email message to specified folder. If no folder is specified, it is saved to project folder.
- Files in existing folder with the same name will be overwritten.

ii. Save Attachments
- Saves the mail message attachments to specified folder. If no folder is specified, it is saved to the project folder.
- Files in existing folder with the same name will be overwritten.

iii. Retrieving unread emails
- Get Outlook Mail Messages & Get IMAP Mail Messages.

iv. Sending email messages
- Send Outlook Mail Message & Send SMTP Mail Message

More resources at:
https://studio.uipath.com/
https://activities.uipath.com/
https://forum.uipath.com/

Debugging

Functions of debugging are located in the Execute tab.

Various functions for identifying and removing errors in a project.

i. Break
- Pause the debugging process at any given moment.
- Available when debugging is in progress.

ii. Step Into
- Allows us to analyse our activities step-by-step.
- Opens & highlights containers.
- Available when debugging is paused.

iii. Step Over
- Debugs the next activity after the current container.
- Highlights containers without opening them.
- Available when debugging is paused.

iv. Validate
- Ensures all variables, arguments, & imports are properly configured & used across the workflow.
- Should be one of the first steps before execution of the workflow.

v. Breakpoints
- Points to pause the debugging process on an activity which may trigger execution issues.
- Can be created from the Execution tab or Context Menu

vi. Slow Step
- Allows us to take a closer look at any activity during debugging at four different available speeds.

vii. Options
- Allows us to focus on fragile parts in our workflow, as such, having UI elements highlighted during debugging or activities logged into the Output Panel.

vii. Log Message / Write Line / Message Box
- These activities can also be used to show the output of our workflows, value of our variables & arguments.

With compliments from:

CFB bots

Keyboard Shortcuts

Some keyboard shortcuts for various activities to save time.

i. File Management
- **Ctrl + Shift + N** (Create new blank process)
- **Ctrl + O** (Open previously created workflows)
- **Ctrl + L** (Open Log files folder)
- **Ctrl + S** (Save currently opened workflow)
- **Ctrl + Shift + S** (Save all opened workflows)

ii. Comments
- **Ctrl + D** (Ignore an activity by placing it in a Comment Out container)
- **Ctrl + E** (Remove an activity placed in a Comment Out container)

iii. Debugging
- **F7** (Runs currently opened workflow in debug mode)
- **F8** (Checks currently opened workflow for validation errors)
- **F9** (Mark selected activity with a breakpoint)
- **Shift + F9** (Removes all breakpoints in the currently opened workflow)
- **F11** (During debugging, Step Into function)
- **Shift + F11** (During debugging, Step Over function)

iv. Recording
- **Alt + Ctrl + W** (Opens Web recording toolbar)
- **Alt + Ctrl + B** (Opens Basic recording toolbar)
- **Alt + Ctrl + C** (Opens Citrix recording toolbar)
- **Alt + Ctrl + D** (Opens Desktop recording toolbar)
- **F2** (Add delay while recording)
- **F3** (Specify a custom recording region)
- **F4** (Choose UI Framework to record with, Default/AA/UIA)

v. Workflow Execution
- **F5** (Runs currently opened workflow)
- **F12** (Stops execution of current workflow)

vi. Selected Activity
- **Ctrl + T** (Places activity inside a Try section of Try-Catch activity)
- **Ctrl + N** (Creates a new Sequence Diagram)
- **Ctrl + C** (Copy selected activity)
- **Ctrl + V** (Pasted copied activity)

Output methods for Screen Scraping

Methods	Speed	Accuracy	Background	Text Position	Hidden Text	Citrix
Full Text	100%	100%	Yes	No	Yes	No
Native	80%	100%	No	Yes	No	No
OCR	30%	98%	No	Yes	No	Yes