Welcome to pyOpenRPA's wiki

by Ivan Maslov (Russia) - see 3. Copyrights & Contacts.

! ATTENTION ! pyOpenRPA works only on MS Windows 7+/Server 2008+. Guys from Unix/Mac - sorry. We will come to you soon :)

Donate

pyOpenRPA is absolutely non-commercial project.

Please donate some $ if pyOpenRPA project is actual for you. Link to online donations.
https://yoomoney.ru/to/4100115560661986

About

Dear RPA-tors. Let me congratulate you with great change in the RPA world. Since 2019 the first enterprise level open source RPA platform is here!

The pyOpenRPA - free, fast and reliable Powerful OpenSource RPA tool for business (based on python 3). Best performance and absolutely free!

The pyOpenRPA is based on Python and using well known OpenSource solutions such as Selenium, OpenCV, Win32, UI automation and others. Thanks to it we were able to create consolidated platform with all possible features. The pyOpenRPA is distributed under the MIT license which allows you to use it in any way you want and any time you need without any restrictions. At the time of this writing the pyOpenRPA is successfully using in several big Russian companies. Companies in which it was decided to develop own RPA division with no dependencies on expensive licenses.

Repo structure

The description of the each folder in GitLab repo is going below:

- **Agent**: template build for the pyOpenRPA.Agent component
- **Orchestrator**: template build for the pyOpenRPA.Orchestrator component
- **Resources**: 3rd party resources which is needed to provide pyOpenRPA encapsulation from the Operating System (OS) dependencies.
- **Robot**: template build for the robot
- **Sources**: pyOpenRPA python package sources + sphinx sources

- **Studio**: build for the pyOpenRPA.Studio which support Desktop UI (support x32 and x64 desktop UI apps)
- **Utils**: many additional good tools for the python developer
- **Wiki**: compiled wiki documentation

The pyOpenRPA structure

The pyOpenRPA has 4 main tools:

- Studio
- Robot
- Orchestrator
- Agent

Studio

The pyOpenRPA.Studio tool has been developed to help RPA-tors to create the robot algorythms.

Features

- Run actions
- Create visual algorythms of the robot
- Desktop app: Analyze desktop app ui tree
- Desktop app: Search desktop app ui by mouse
- Desktop app: Generate & edit the UIO Selector

Robot

The pyOpenRPA.Robot package is the core of any action execution in the pyOpenRPA. All action from algorythms are perfoming by the Robot tool. It looks like a console process without graphic user interface.

Features

- Based on Python (killer feature)
- Support Win32 GUI framework (desktop app)
- Support UI automation framework (desktop app)
- Support Selenium (web app)
- Support PyAutoGUI (screen capture & mouse)
- Support OpenCV (computer vision)

Orchestrator

The pyOpenRPA.Orchestrator package has been developed to maintain robot infrastructure (2+ robots algorythm).

Features

- Start/Stop robot algorithm
- Robot scheduler
- Remote machine screenshot viewer
- Remote machine cmd shell
- Remote machine logs storage

Agent

The pyOpenRPA.Agent tool has been developed to maintain robot infrastructure (2+ robots algorythm).

Features

- Send CMD to the RPA GUI session (start/safe stop/force stop/logout)
- Get screenshots from the RPA GUI session
- Get the list of the running processes

Wiki structure

In wiki you can use the following docs:

- ENG Guide HTML [|OPEN GITLAB|]
- ENG Guide MarkDown [|OPEN GITLAB|]
- ENG Guide PDF [|OPEN GITLAB|]
- RUS Article: Less cost - no paid RPA [|OPEN HABR|]
- RUS Tutorial Desktop UI [|OPEN HABR|]; [|OPEN GITLAB|]
- RUS Tutorial Web UI [|OPEN HABR|]; [|OPEN GITLAB|]
- RUS Leaflet pyOpenRPA v4.pdf [|OPEN GITLAB|]

Guide content

GENERAL

ROBOT

STUDIO

ORCHESTRATOR

AGENT

Next

1. How to install

Are you ready to install the pyOpenRPA solution on your machine?

Ok, we start. **Do the following operations:**

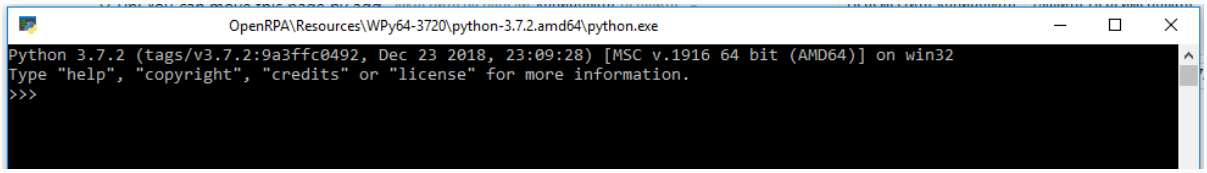
- Download the pyOpenRPA package from master branch on GitLab [Download ZIP] (https://gitlab.com/UnicodeLabs/OpenRPA/-/archive/master/OpenRPA-master.zip)
- Unzip the package

Installation has been completed :)

How to check installation

- Run portable python (built in the pyOpenRPA)
  - x32 python (GIT\Resources\WPy32-3720\python-3.7.2\python.exe)
  - x64 python (GIT\Resources\WPy64-3720\python-3.7.2.amd64\python.exe)

The pyOpenRPA has been successfully installed if the portable python 3.7.2 was started without any exceptions (see screenshot).



System requirements

- OS Windows 7+. *Need Windows package KB2999226 if use windows Vista/7/8/8.1/Server 2008/Server 2012 (https://support.microsoft.com/ru-ru/help/2999226)*
- For OpenCV: OS Windows 7/8/8/10 only (no Windows Server)

Previous

Next

2. Roadmap

- **Guide**
  - **ENG - done 2021.03.11**
    - HTML |OPEN GITLAB|
    - MarkDown |OPEN GITLAB|
    - PDF [|OPEN GITLAB|]
  - RUS - queue
- **Tutorial**
  - ENG - queue
  - **RUS - in progress**
    - Article: Less cost - no paid RPA |OPEN HABR|
    - Tutorial Desktop UI |OPEN HABR|; |OPEN GITLAB|
    - Tutorial Web UI |OPEN HABR|; |OPEN GITLAB|
    - Article: RPA as a core of the IT automation - soon
- **Leaflet**
  - ENG queue
  - **RUS done 2021.02.23**
    - RUS Leaflet pyOpenRPA v4.pdf |OPEN GITLAB|

Previous Next

3. Copyrights & Contacts

pyOpenRPA is created by Ivan Maslov (Russia). Use it absolutely for free!

My purpose is to create #IT4Business models in the companies. I can help you to create the new #IT4Business in your company. #IT4Business homepage - https://www.facebook.com/RU.IT4Business #IT4Busines is the methodology which is created for build compact fast and reliable IT function in company. If you has many IT specialists, very long deadlines for the IT tasks, many bugs in IT software - #IT4Business is for you :)

If you need some IT help - feel free to contact me (prefer e-mail or skype). If you will find some issue in pyOpenRPA - write about it to me via e-mail/skype/gitlab issue.

Thank you!

Ivan Maslov (founder)

- E-mail: Ivan.Maslov@UnicodeLabs.ru
- Skype: MegaFinder
- Facebook: https://www.facebook.com/RU.IT4Business
- LinkedIn: https://www.linkedin.com/in/RU-IvanMaslov/
- WhatsApp | Telegram: +7 906 722 39 25

3-rd party components license dependencies

- WinPython 3.7.1 32-bit & 64-bit, license MIT (https://github.com/winpython/winpython)
- Selenium v..., license Apache 2.0
- pywinauto 0.6.5, license BSD 3-Clause (https://github.com/pywinauto/pywinauto)
- Semantic UI ..., license MIT (https://github.com/Semantic-Org/Semantic-UI)
- PyAutoGUI ..., license BSD 3-Clause (https://github.com/asweigart/pyautogui)
- keyboard ..., license MIT (https://github.com/boppreh/keyboard)
- pywin32

1. Description

pyOpenRPA Robot is the python package which allow you to create best RPA program.

The description of the functions you can find page ‘Defs’ (see menu)

Here is the example of the pyOpenRPA usage.

```
# EXAMPLE 1
from pyOpenRPA.Robot import UIDesktop

lNotepadOKButton = UIDesktop.UIOSelector_Get_UIO(
    inSpecificationList=[
        {"title":"notepad.exe"},{"title":"OK"}],
    inElement=None,
    inFlagRaiseException=True)

lNotepadOKButton.click()
```

Previous

Next

2. Defs

Here you can find the functions description for interaction with desktop GUI applications

How to use both x32 and x64 python processes (it can be helpfully, if another app GUI is on another bitness than your app)

```
from pyOpenRPA.Robot import UIDesktop
#Section for robot init
IPyOpenRPA_SettingsDict = {
    "Python32FullPath": "..\\Resources\\WPy32-3720\\python-3.7.2\\python.exe", #Set from user: "..\\Resources\\WPy32-3720\\python-3.7
    "Python64FullPath": "..\\Resources\\WPy64-3720\\python-3.7.2.amd64\\python.exe", #Set from user
    "Python32ProcessName": "pyOpenRPA_UIDesktopX32.exe", #Config set once
    "Python64ProcessName": "pyOpenRPA_UIDesktopX64.exe" #Config set once
}
# Init the pyOpenRPA configuration
UIDesktop.Utils.ProcessBitness.SettingsInit(IPyOpenRPA_SettingsDict)
# Now you can use pyOpenRPA with both bitness.
```

Functions:

Get_OSBitnessInt ()	Detect OS bitness.
PWASpecification_Get_PWAApplication (...)	#Backend selection - attribute “backend” (“win32” || “uia”) in 1-st list element
PWASpecification_Get_UIO (...)	#Backend def selection - attribute “backend” (“win32” || “uia”) in 1-st list element #old name - GetControl
UIOSelectorSecs_WaitAppear_Bool (...)	Wait for UI object will appear in GUI for inWaitSecs seconds.
UIOSelectorSecs_WaitDisappear_Bool (...)	Wait for UI object will disappear in GUI for inWaitSecs seconds.
UIOSelectorUIOActivity_Run_Dict (...[, ...])	Run the activity in UIO (UI Object)
UIOSelector_Exist_Bool (inUIOSelector)	Check if object is exist by the UIO selector.
UIOSelector_FocusHighlight (inUIOSelector)	Set focus and highlight (draw outline) the element (in app) by the UIO selector.
UIOSelector_GetChildList_UIOList ([...])	Get list of child UIO’s by the parent UIOSelector
UIOSelector_Get_BitnessInt (inSpecificationList)	Detect process bitness by the UI Object UIO Selector.
UIOSelector_Get_BitnessStr (inSpecificationList)	Detect process bitness by the UI Object UIO Selector.
UIOSelector_Get_UIO (inSpecificationList[, ...])	Get the pywinauto object by the UIO selector.
UIOSelector_Get_UIOActivityList (inUIOSelector)	Get the list of the UI object activities
UIOSelector_Get_UIOInfo (inUIOSelector)	Get the UIO dict of the attributes
UIOSelector_Get_UIOList (inSpecificationList)	Get the UIO list by the selector
UIOSelector_Highlight (inUIOSelector)	Highlight (draw outline) the element (in app) by the UIO selector.

UIOSelector_SafeOtherGet_Process (inUIOSelector)	Safe get other process or None if destination app is the other/same bitness
UIOSelector_SearchChildByMouse_UIO (...)	UIOSelector (see description on the top of the document) #old name - AutomationSearchMouseElement
UIOSelector_SearchChildByMouse_UIOTree (...)	!!!!Safe call is included (you can set activity and UIDesktop will choose the bitness and return the result)!!!!!
UIOSelector_TryRestore_Dict (inSpecificationList)	Try to restore (maximize) window, if it's minimized.
UIOSelectorsSecs_WaitAppear_List (...[, ...])	Wait for many UI object will appear in GUI for inWaitSecs seconds.
UIOSelectorsSecs_WaitDisappear_List (...[, ...])	Wait for many UI object will disappear in GUI for inWaitSecs seconds.

pyOpenRPA.Robot.UIDesktop.Get_OSBitnessInt() [source]

Detect OS bitness.

Returns:
int 32 or int 64

pyOpenRPA.Robot.UIDesktop.PWASpecification_Get_PWAApplication(*inControlSpecificationArray*) [source]

#Backend selection - attribute "backend" ("win32" || "uia") in 1-st list element

Parameters:
inControlSpecificationArray – List of dict, dict in pywinauto.find_windows notation

Returns:
process application object

pyOpenRPA.Robot.UIDesktop.PWASpecification_Get_UIO(*inControlSpecificationArray*) [source]

#Backend def selection - attribute "backend" ("win32" || "uia") in 1-st list element #old name - GetControl

Parameters:
inControlSpecificationArray – List of dict, dict in pywinauto.find_windows notation

Returns:
list of UIO object

pyOpenRPA.Robot.UIDesktop.UIOSelectorSecs_WaitAppear_Bool(*inSpecificationList, inWaitSecs*) [source]

Wait for UI object will appear in GUI for inWaitSecs seconds.

Parameters:
- **inSpecificationList** – UIOSelector. Example: [{"title":"notepad"},{"title":"OK"}]
- **inWaitSecs** – Float value (seconds) for wait UI element appear in GUI

Returns:
True - UI object will appear. False - else case

pyOpenRPA.Robot.UIDesktop.UIOSelectorSecs_WaitDisappear_Bool(*inSpecificationList, inWaitSecs*) [source]

Wait for UI object will disappear in GUI for inWaitSecs seconds.

Parameters:
- **inSpecificationList** – UIOSelector. Example: [{"title":"notepad"},{"title":"OK"}]
- **inWaitSecs** – Float value (seconds) for wait UI element disappear in GUI

Returns:
True - UI object will disappear. False - else case

pyOpenRPA.Robot.UIDesktop.UIOSelectorUIOActivity_Run_Dict(*inUIOSelector*, *inActionName*, *inArgumentList=None*, *inkwArgumentObject=None*) [source]

Run the activity in UIO (UI Object)

Parameters:
- **inUIOSelector** – UIOSelector - List of items, which contains condition attributes
- **inActionName** – UIOActivity (name) activity name string from Pywinauto
- **inArgumentList** –
- **inkwArgumentObject** –

Returns:

pyOpenRPA.Robot.UIDesktop.UIOSelector_Exist_Bool(*inUIOSelector*) [source]

Check if object is exist by the UIO selector.

Parameters:
inUIOSelector –

Returns:
True - Object is exist. False - else case

pyOpenRPA.Robot.UIDesktop.UIOSelector_FocusHighlight(*inUIOSelector*) [source]

Set focus and highlight (draw outline) the element (in app) by the UIO selector.

Parameters:
inUIOSelector – UIOSelector - List of items, which contains condition attributes

Returns:

pyOpenRPA.Robot.UIDesktop.UIOSelector_GetChildList_UIOList(*inUIOSelector=None*, *inBackend='win32'*) [source]

Get list of child UIO's by the parent UIOSelector

Parameters:
- **inUIOSelector** – UIOSelector - List of items, which contains condition attributes
- **inBackend** – "win32" or "uia"

Returns:

pyOpenRPA.Robot.UIDesktop.UIOSelector_Get_BitnessInt(*inSpecificationList*) [source]

Detect process bitness by the UI Object UIO Selector.

Parameters:
inSpecificationList – UIOSelector. Example: [{"title":"notepad"},{"title":"OK"}]

Returns:
int 32 or int 64

pyOpenRPA.Robot.UIDesktop.UIOSelector_Get_BitnessStr(*inSpecificationList*) [source]

Detect process bitness by the UI Object UIO Selector.

Parameters:

inSpecificationList – UIOSelector. Example: [{“title”:”notepad”},{“title”:”OK”}]

Returns:
str “32” or str “64”

---

pyOpenRPA.Robot.UIDesktop.UIOSelector_Get_UIO(*inSpecificationList, inElement=None, inFlagRaiseException=True*) [source]

Get the pywinauto object by the UIO selector.

Parameters:
- **inSpecificationList** –
- **inElement** –
- **inFlagRaiseException** –

Returns:

---

pyOpenRPA.Robot.UIDesktop.UIOSelector_Get_UIOActivityList(*inUIOSelector*) [source]

Get the list of the UI object activities

Parameters:
inUIOSelector – UIOSelector - List of items, which contains condition attributes

Returns:

---

pyOpenRPA.Robot.UIDesktop.UIOSelector_Get_UIOInfo(*inUIOSelector*) [source]

Get the UIO dict of the attributes

Parameters:
inUIOSelector – UIOSelector - List of items, which contains condition attributes

Returns:

---

pyOpenRPA.Robot.UIDesktop.UIOSelector_Get_UIOList(*inSpecificationList, inElement=None, inFlagRaiseException=True*) [source]

Get the UIO list by the selector

Parameters:
- **inSpecificationList** – UIO Selector
- **inElement** – Входной элемент - показатель, что не требуется выполнять коннект к процессу
- **inFlagRaiseException** – Флаг True - выкинуть ошибку в случае обнаружении пустого списка

Returns:

---

pyOpenRPA.Robot.UIDesktop.UIOSelector_Highlight(*inUIOSelector*) [source]

Highlight (draw outline) the element (in app) by the UIO selector.

Parameters:
inUIOSelector – UIOSelector - List of items, which contains condition attributes

Returns:

---

pyOpenRPA.Robot.UIDesktop.UIOSelector_SafeOtherGet_Process(*inUIOSelector*) [source]

Safe get other process or None if destination app is the other/same bitness

Parameters:
inUIOSelector – UIO Selector of the UI object

Returns:
None or process (of the other bitness)

pyOpenRPA.Robot.UIDesktop.UIOSelector_SearchChildByMouse_UIO(*inElementSpecification*) [source]

UIOSelector (see description on the top of the document) #old name - AutomationSearchMouseElement

Parameters:
inElementSpecification – UIOSelector of the UI Object

Returns:
pywinauto element wrapper instance or None

pyOpenRPA.Robot.UIDesktop.UIOSelector_SearchChildByMouse_UIOTree(*inUIOSelector*) [source]

!!!!Safe call is included (you can set activity and UIDesktop will choose the bitness and return the result)!!!!!

Parameters:
inUIOSelector – UIOSelector of the UI Object

Returns:

?

pyOpenRPA.Robot.UIDesktop.UIOSelector_TryRestore_Dict(*inSpecificationList*) [source]

Try to restore (maximize) window, if it's minimized. (!IMPORTANT! When use UIA framework minimized windows doesn't appear by the UIOSelector. You need to try restore windows and after that try to get UIO)

Parameters:
inSpecificationList – UIOSelector - List of items, which contains condition attributes

Returns:

pyOpenRPA.Robot.UIDesktop.UIOSelectorsSecs_WaitAppear_List(*inSpecificationListList, inWaitSecs, inFlagWaitAllInMoment=False*) [source]

Wait for many UI object will appear in GUI for inWaitSecs seconds.

Parameters:

- **inSpecificationListList** –
  UIOSelector list. Example: [
  [{"title":"notepad"},{"title":"OK"}], [{"title":"notepad"},{"title":"Cancel"}]
  ]
- **inWaitSecs** – Float value (seconds) for wait UI element appear in GUI
- **inFlagWaitAllInMoment** – True - Wait all UI objects from the UIOSelector list to be appeared

Returns:
List of index, which UI object UIO will be appeared. Example: [1] # Appear only UI object with UIO selector: [{"title":"notepad"},{"title":"Cancel"}]

pyOpenRPA.Robot.UIDesktop.UIOSelectorsSecs_WaitDisappear_List(*inSpecificationListList, inWaitSecs, inFlagWaitAllInMoment=False*) [source]

Wait for many UI object will disappear in GUI for inWaitSecs seconds.

Parameters:

- **inSpecificationListList** –

UIOSelector list. Example: [
[{“title”:”notepad”},{“title”:”OK”}], [{“title”:”notepad”},{“title”:”Cancel”}]
]

- **inWaitSecs** – Float value (seconds) for wait UI element disappear in GUI
- **inFlagWaitAllInMoment** – True - Wait all UI objects from the UIOSelector list to be disappeared.

Returns:
List of index, which UI object UIO will be disappeared. Example: [1] # Disappear only UI object with UIO selector: [{“title”:”notepad”},{“title”:”Cancel”}]

Returns:

References

reStructuredText [1]

[1] :
http://docutils.sourceforge.net/rst.html

Previous Next

3. How to use

The Robot tool is the main module for production process automation. It has no graphic/console interface. All low-level actions to OS are perfoming by the Robot tool in pyOpenRPA.

How to execute RPA script

You can use the robot by the several ways:

- In Python script
- In Studio script (n/a)

Create python script

In order to use robot just add Robot tool folder in work directory and add line “import GUI” in your script.

```
import sys
sys.path.append('../../')
import selenium # [Web app access](https://gitlab.com/UnicodeLabs/OpenRPA/wikis/05.1.-Theory-&-practice:-Web-app-access-(Chron
import GUI  # [Win32 & UI Automation access](https://gitlab.com/UnicodeLabs/OpenRPA/wikis/05.2.-Theory-&-practice:-Desktop-app-U
import pyautogui #[Screen capture/recognition](https://gitlab.com/UnicodeLabs/OpenRPA/wikis/05.4.-Theory-&-practice:-Screen-captur
import cv2 # [Computer vision](https://gitlab.com/UnicodeLabs/OpenRPA/wikis/05.4.-Theory-&-practice:-Screen-capture-&-image-recog
import keyboard #[Keyboard manipulation](https://gitlab.com/UnicodeLabs/OpenRPA/wikis/05.3.-Theory-&-practice:-Keyboard-&-mous
```

Execute python script

The pyOpenRPA is fully portable solution. It contains own python enviroment both 32 and 64 bit versions. So, you can execute your python script in several ways: - Execute in python x32 (OpenRPAResourcesWPy32-3720python-3.7.2) - Execute in python x64 (OpenRPAResourcesWPy64-3720python-3.7.2.amd64) - Execute from .cmd file

Execute in the Python x32

To execute your python script in x32 bit version just write in command line from x32 python directory:

```
cd "\OpenRPA\Resources\WPy32-3720\python-3.7.2"
python.exe "path to your python script.py"
```

Execute in the Python x64

To execute your python script in x32 bit version just write in command line from x32 python directory:

```
cd "\OpenRPA\Resources\WPy64-3720\python-3.7.2.amd64"
python.exe "path to your python script.py"
```

Execute from .cmd file

In order to simplify the execution process you can write several code lines in file with the .cmd extension:

```
cd %~dp0
copy /Y ..\Resources\WPy32-3720\python-3.7.2\python.exe ..\Resources\WPy32-3720\python-3.7.2\OpenRPAOrchestrator.exe
.\..\Resources\WPy32-3720\python-3.7.2\OpenRPAOrchestrator.exe orchestratorMain.py
pause >nul
```

Use in studio script (n/a)

```
import sys
sys.path.append('../../')
import GUI
import keyboard
import subprocess
import time

#Highlight the UI Object in Folder explorer
GUI.UIOSelector_FocusHighlight([{"class_name":"CabinetWClass","backend":"uia"},{"ctrl_index":2},{"ctrl_index":0},{"ctrl_index":2},{"ctrl_i

#Wait 2 seconds
time.sleep(3)

#Loop: get child element of UI List
for lItem in GUI.UIOSelector_Get_UIO([{"class_name":"CabinetWClass","backend":"uia"},{"ctrl_index":2},{"ctrl_index":0},{"ctrl_index":2},{"
    print(str(lItem))
```

Here you can find the docs and examples of the OpenRPA desktop (GUI) app access.

Desktop app UI access (win32 and UI automation dlls)

Definitions

- **UIO** - UI Object (class of pywinauto UI object) [pywinauto.base_wrapper]
- **UIOSelector** - List of dict (key attributes)
- **PWA** - PyWinAuto
- **PWASpecification** - List of dict (key attributes in pywinauto.find_window notation)
- **UIOTree** - Recursive Dict of Dict ... (UI Parent -> Child hierarchy)
- **UIOInfo** - Dict of UIO attributes
- **UIOActivity** - Activity of the UIO (UI object) from the Pywinauto module
- **UIOEI** - UI Object info object

What is UIO?

UIO is a User Interface Object (pyOpenRPA terminology). For maximum compatibility, this instance is inherited from the object model developed in the [pywinauto library (click to get a list of available class functions)](https://pywinauto.readthedocs.io/en/latest/code/pywinauto.base_wrapper.html).

This approach allows us to implement useful functionality that has already been successfully developed in other libraries, and Supplement it with the missing functionality. In our case, the missing functionality is the ability to dynamically access UIO objects using UIO selectors.

UIOSelector structure & example

UIOSelector is the list of condition items for the UIO in GUI. Each item has condition attributes for detect applicable UIO. Here is the description of the available condition attributes in item.

Desciption

```
[
  {
    "depth_start" :: [int, start from 1] :: the depth index, where to start check the condition list (default 1),
    "depth_end" :: [int, start from 1] :: the depth index, where to stop check the condition list (default 1),
    "ctrl_index" || "index" :: [int, starts from 0] :: the index of the UIO in parent UIO child list,
    "title" :: [str] :: the condition for the UIO attribute *title*,
    "title_re" :: [str] :: regular expression (python ver) for the condition for the UIO attribute *title*,
    "rich_text" :: [str] :: the condition for the UIO attribute *rich_text*,
    "rich_text_re" :: [str] :: regular expression (python ver) for the condition for the UIO attribute *rich_text*,
    "class_name" :: [str] :: the condition for the UIO attribute *class_name*,
    "class_name_re" :: [str] :: regular expression (python ver) for the condition for the UIO attribute *class_name*,
    "friendly_class_name" :: [str] :: the condition for the UIO attribute *friendly_class_name*,
    "friendly_class_name_re" :: [str] :: regular expression (python ver) for the condition for the UIO attribute *friendly_class_name*,
    "control_type" :: [str] :: the condition for the UIO attribute *control_type*,
    "control_type_re" :: [str] :: regular expression (python ver) for the condition for the UIO attribute *control_type*,
    "is_enabled" :: [bool] :: the condition for the UIO attribute *is_enabled*. If UI object is enabled on GUI,
    "is_visible" :: [bool] :: the condition for the UIO attribute *is_visible*. If UI object is visible on GUI,
    "backend" :: [str, "win32" || "uia"] :: the method of UIO extraction (default "win32"). ATTENTION! Current option can be only for the first
  },
  { ... specification next level UIO }
]
```

The UIO selector example

```
[
  {"class_name":"CalcFrame", "backend":"win32"}, # 1-st level UIO specification
  {"title":"Hex", "depth_start":3, "depth_end": 3} # 3-rd level specification (because of attribute depth_start|depth_stop)
]
```

The UIDesktop module (OpenRPA/Robot/UIDesktop.py)

The UIDesktop is extension of the pywinauto module which provide access to the desktop apps by the **win32** and **ui automation** dll frameworks (big thx to the Microsoft :)).

```
# EXAMPLE 1
from pyOpenRPA.Robot import UIDesktop

UIDesktop.UIOSelector_Get_UIO(
  inSpecificationList=[
    {"title":"notepad.exe"},{"title":"OK"}],
  inElement=None,
  inFlagRaiseException=True)
```

The UIDesktop module (OpenRPA/Robot/UIDesktop.py)

The UIDesktop is extension of the pywinauto module which provide access to the desktop apps by the **win32** and **ui automation** dll frameworks (big thx to the Microsoft :)).

Naming convention: <InArgument>_<ActivityName>_<OutArgument - if exist>

Theory & practice. WEB app UI access (selenium)

About

The pyOpenRPA support web app manipulation (by the Selenium lib). More docs about selenium you can find here (https://selenium-python.readthedocs.io/)

How to use

To start use selenium just import selenium modules in the robot tool. Here is the example of the usage.

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys

driver = webdriver.Chrome()
driver.get("http://www.python.org")
assert "Python" in driver.title
elem = driver.find_element_by_name("q")
elem.clear()
elem.send_keys("pycon")
elem.send_keys(Keys.RETURN)
assert "No results found." not in driver.page_source
driver.close()
```

Theory & practice. Keyboard & mouse manipulation

Theory & practice. Screen capture & image recognition

How to automate image recognition on PC

Here you can find any ways you need to use in your business case: - Find the exact match on the screen with the other image - Use text recognition module (OCR) - Use computer vision (CV) to identify the objects on screen/image/video - Use artificial intelligence (AI) to make custom identification/classification/text recognition

Previous Next

4. Dependencies

Python 3 x32 [psutil, pywinauto, wmi, PIL, keyboard, pyautogui, win32api (pywin32), selenium, openCV, tesseract, requests, lxml, PyMuPDF] Python 3 x64 [psutil, pywinauto, wmi, PIL, keyboard, pyautogui, win32api (pywin32), selenium, openCV, tesseract, requests, lxml, PyMuPDF] pywinauto (Windows GUI automation) Semantic UI CSS framework JsRender by https://www.jsviews.com (switch to Handlebars) Handlebars

Previous

Next

1. Description

pyOpenRPA Studio is the executable process.

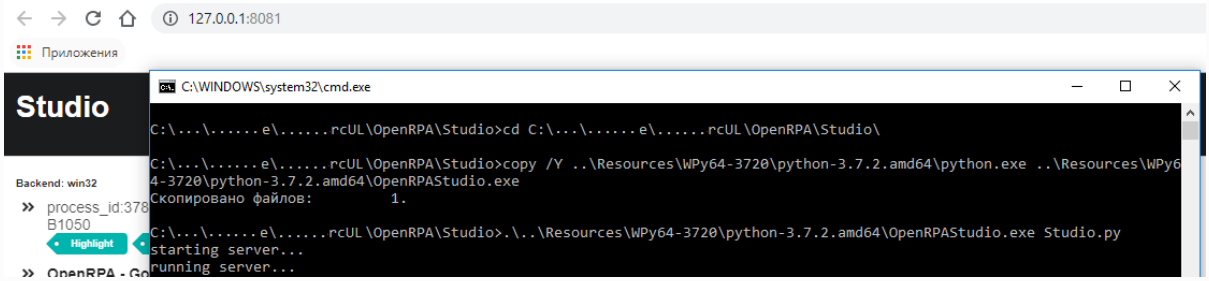
Previous Next

2. How to use

Content

How to run

- For OS x32
- Run (double click): OpenRPA_32.cmd (for OS x32)
- For OS x64
- Run (double click): OpenRPA_64.cmd (for OS x64)
- Wait text “running server” in console. Default browser will be open automatically
- **Attention!** The studio tool does not support the Internet explorer (any version) for GUI rendering (lol)



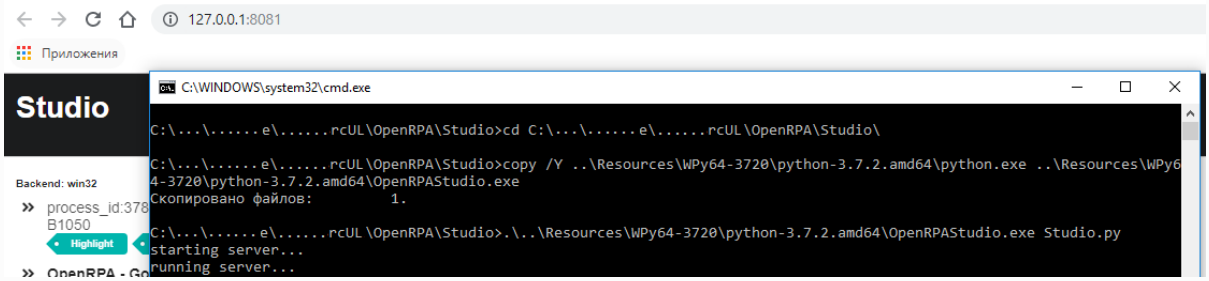
UI Description

The studio tool GUI contains of:

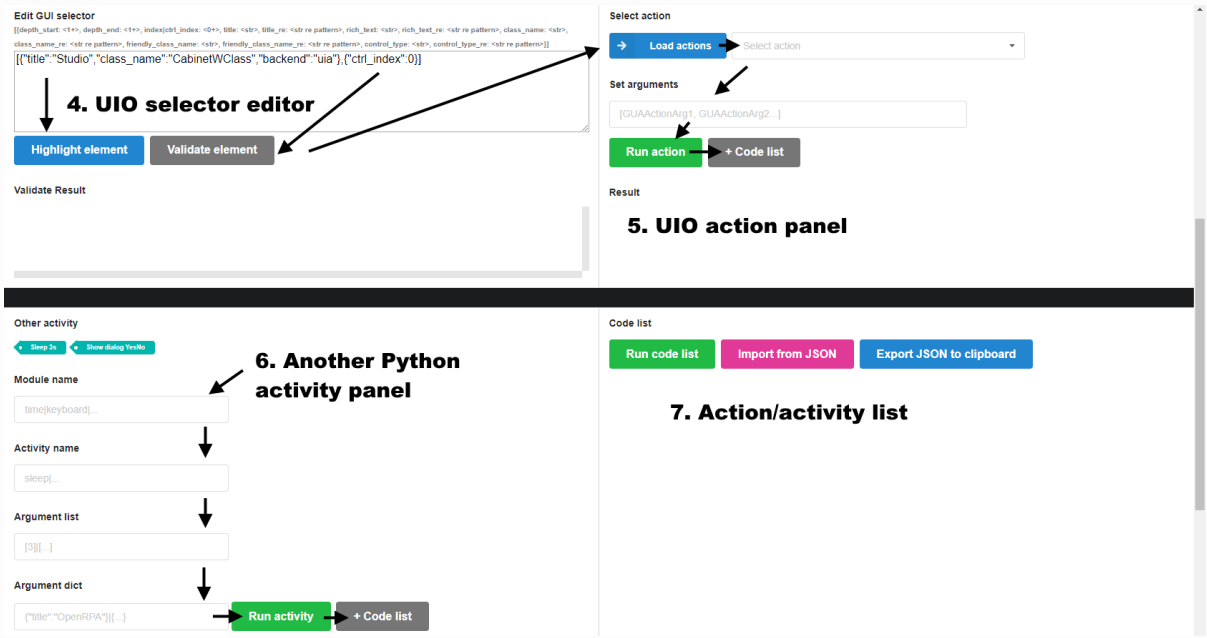
1. UI tree viewer
2. Selected UI object hierarchy list
3. Selected UI object property list
4. UIO selector editor
5. UIO action panel
6. Another Python activity panel
7. Action/activity list

Look it on the GUI screenshots are listed below

GUI Screenshot 1



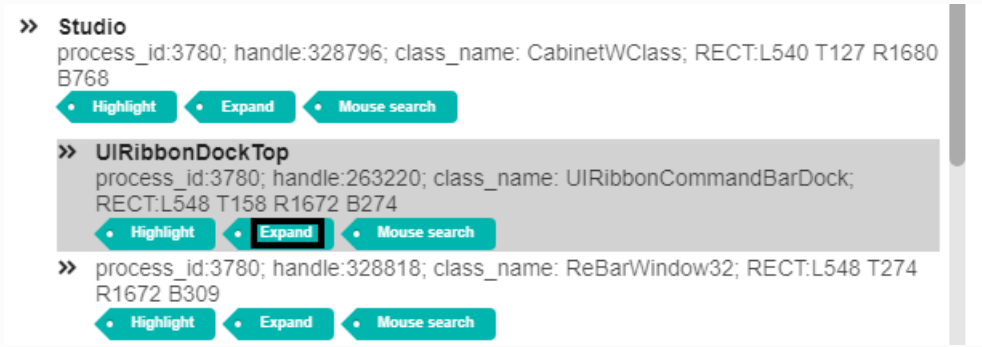
GUI Screenshot 2



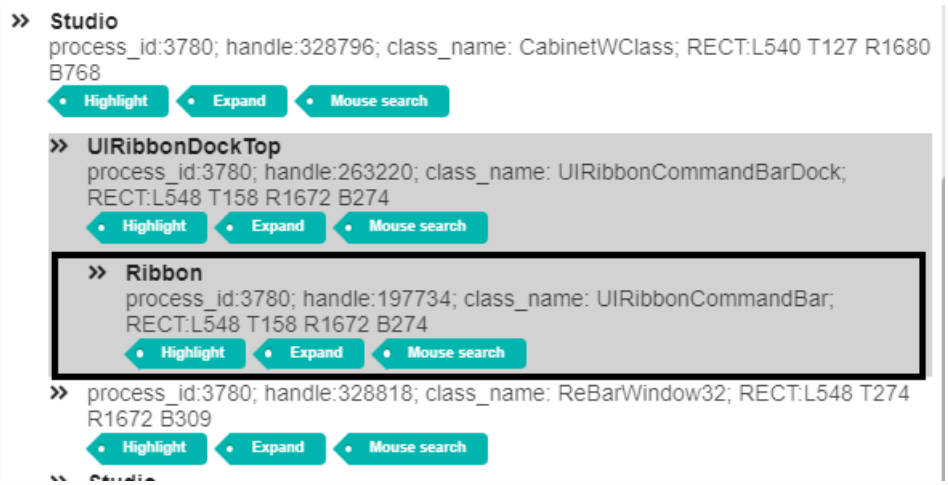
How to extract UI tree

In order to extract the UI tree do the following: in UI tree viewer choose the object you are interested and click the button “Expand”. ##

Action: Click the button “Expand”



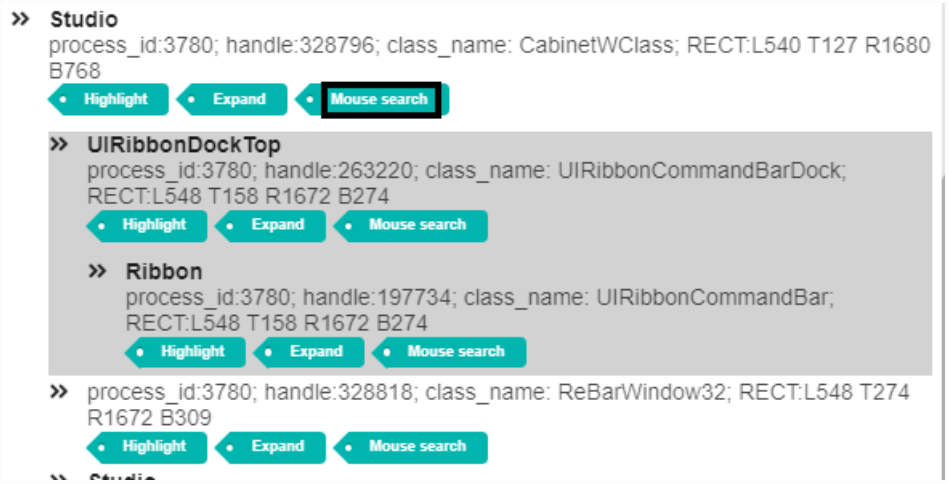
Result



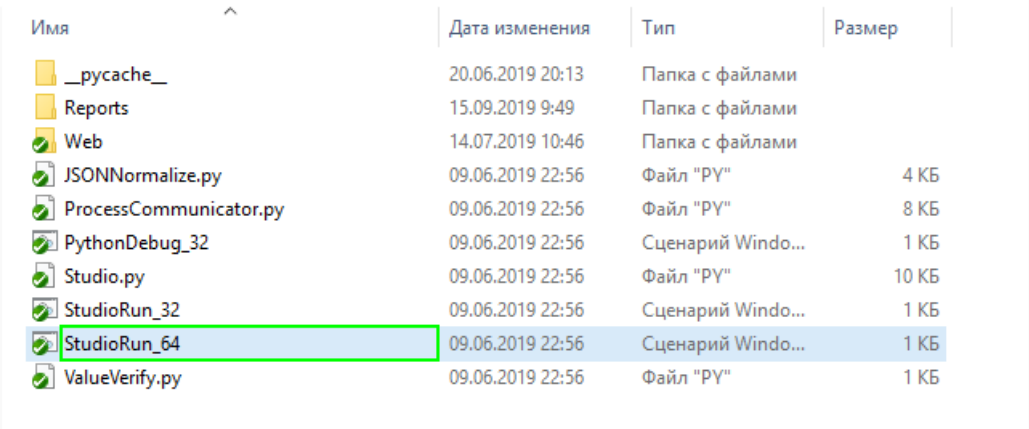
How to search UI object by mouse hover

In order to search UI object do the following: in UI tree viewer choose the parent object, where you are want to search UI object, and click the button “Mouse search”. The mouse search mode will start. Turn mouse on the UI object you are interested and wait when the studio will highlight the UI object. After the hightlight hold the “Ctrl” key and wait 3 seconds. The interested UI object will be shown in UI tree viewer.

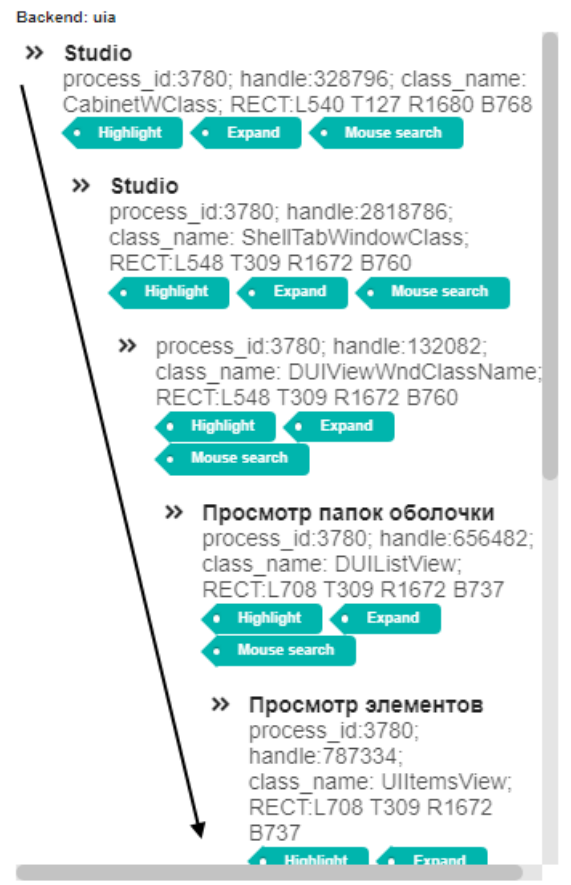
Action: Click the button “Mouse search”



Action: Turn mouse on the UI object you are interested and hold the “Ctrl” key for 3 seconds



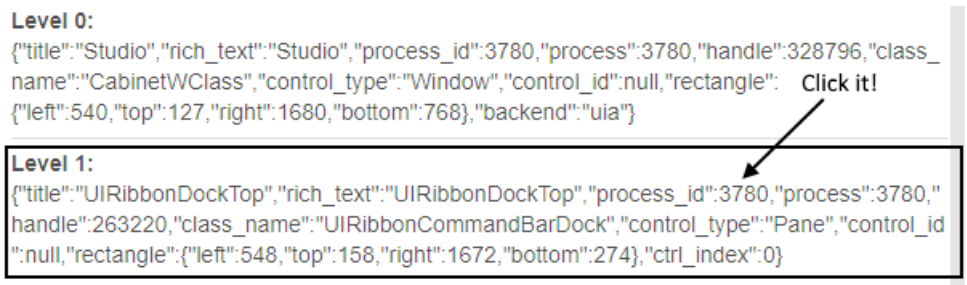
Result: The interested UI object will be shown in UI tree viewer



How to extract UI object properties

In order to extract UI object properties do the following: in Selected UI object hierarchy list choose the UI object you are interested and click it. The UI object property list will be shown in Selected UI object property list

Action: Choose the UI object you are interested and click it



Result: The UI object property list will be shown in Selected UI object property list

class_name: "UIRibbonCommandBarDock"

friendly_class_name: "Pane"

texts: ["UIRibbonDockTop"]

control_id: 0

is_visible: true

is_enabled: true

control_count: 1

is_keyboard_focusable: true

has_keyboard_focus: false

automation_id: ""

Previous Next

1. Description

pyOpenRPA Orchestrator is the executable process.

The features of the orchestrator is: - Centralized/decentralized user control interface (applicable and for business users and for technical users). Web based, support desktop, tablet, phone. - Automatized robots control (customized algorithms, robots scheduling) - Source code mega flexibility: Light Orchestrator architecture is good for own customization

Global settings dict concept

pyOpenRPA project is complex tool which consist of several executable modules such as Robot, Orchestrator, Studio,

Because of module compexity, we use 1 init arg - inGSettings inGSettings is a complex dictionary which has all reqired parameters for the module execution.

The description of the GSettings you can find in executable module details.

Orchestrator how to configure

To init pyOpenRPA Orchestrator instance use script:

from pyOpenRPA import Orchestrator # Import orchestrator main gSettings = SettingsTemplate.Create(inModeStr="BASIC") # Create GSettings with basic configuration Orchestrator.Orchestrator(inGSettings=gSettings) # Call the orchestrator def

gSettings structure

Orchestrator architecture

Orchestrator has several source code components: - User/robot activity consolidated queue single thread (Processor) - User/robot activity asynchonus many threads (Processor) - Scheduler single thread (main) - RDP keep active many thread - Autocleaner single thread - GUI keep active single thread - HTTP web server single thread (create user socket threads) -

Below you can find more information about all of the component.

Component Processor

Sync - Append activity list to consolidated processor queue. Execution goes sequency by the activity list order

Async - Create New thread to execute the activity list

- Activity list

List of the activity item

- Activity item

Activity item is universal mechanism to execute different algorythms from any sources. The core feature of the Activity is to call python defs with args and kwargs. If you need to init do some activity you can write some python def, then create Activity item with current def. ATTENTION: In some cases (such as web transmition), when you can't transmit python def as object you can use symbolic names for python defs. It is apply you to init all of you want from the web UI.

?Why i cant transmit python def from the web Because the WEB space is not the Python executable space. Interaction between it spaces create by JSON protocol. So, we know than JSON apply int, float, str, bool, None, list, dict - that is all.

Note

Example {

"Def":"DefAliasTest", # def link or def alias (look gSettings["Processor"]["AliasDefDict"]) "ArgList": [1,2,3], # Args list "ArgDict":{"ttt":1,"222":2,"dsd":3}, # Args dictionary "ArgGSettings": None # Name of GSettings attribute: str (ArgDict) or index (for ArgList) "ArgLogger": None # Name of GSettings attribute: str (ArgDict) or index (for ArgList)

}# Pay attention! Do not left comma symbol after the end of the dict - it can be interpretated like a turple..

References

`Python-sphinx`_

2. Defs

pyOpenRPA.Orchestrator.__Orchestrator__

```
# EXAMPLE 1
from pyOpenRPA import Orchestrator
Orchestrator.OSCMD(inCMDStr = "git status", inRunAsyncBool=True)

# EXAMPLE 2
from pyOpenRPA.Orchestrator import __Orchestrator__
__Orchestrator__.OSCMD(inCMDStr = "git status", inRunAsyncBool=True)
```

Group Agent...

Interaction between Orchestrator and pyOpenRPA.Agent daemon process, which can be deployed in another user session.

Group GSettings...

Basic defs to work with singleton gSettings.

Group OS...

Interaction with shell on the Orchestrator user session.

Group Process...

Interaction with some process on the Orchestrator user session.

Group Processor...

Work with Processor queue (see ...).

Group Python...

Work with extra python modules.

Group RDPSession...

Interaction with RDP session, where you can manage some robots.

Group Web...

Manipulate the Orchestrator WEB side.

Group UAC...

Manipulate the User Access Controls (actual for the Orchestrator WEB access for the business users)

Group Scheduler...

Work with activity scheduling.

Functions:

AgentActivityItemAdd (inGSettings, ...)	Add activity in AgentDict.
AgentActivityItemReturnExists (inGSettings, ...)	Check by GUID if ActivityItem has been executed and result has come to the Orchestrator
AgentActivityItemReturnGet (inGSettings, ...)	Work synchroniously! Wait while result will be recieved.
AgentOSCMD (inGSettings, inHostNameStr, ...)	Send CMD to OS thought the pyOpenRPA.Agent daemon.
AgentOSFileBinaryDataBase64StrCreate (...)	Create binary file by the base64 string by the pyOpenRPA.Agent daemon process (safe for JSON transmission)
AgentOSFileBinaryDataBase64StrReceive (...)	Read binary file and encode in base64 to transmit (safe for JSON transmition)
AgentOSFileBinaryDataBytesCreate (...)	Create binary file by the base64 string by the pyOpenRPA.Agent daemon process (safe for JSON transmition)
AgentOSFileTextDataStrCreate (inGSettings, ...)	Create text file by the string by the pyOpenRPA.Agent daemon process
AgentOSFileTextDataStrReceive (inGSettings, ...)	Read text file in the agent GUI session
AgentProcessWOExeUpperUserListGet (...)	Return the process list only for the current user (where Agent is running) without .EXE in upper case.
GSettingsAutocleaner (inGSettings)	HIDDEN Interval gSettings auto cleaner def to clear some garbage.
GSettingsKeyListValueAppend (inGSettings, inValue)	Append value in GSettings by the key list
GSettingsKeyListValueGet (inGSettings[, ...])	Get the value from the GSettings by the key list
GSettingsKeyListValueOperatorPlus (...[, ...])	Execute plus operation between 2 lists (1:inValue and 2:gSettings by the inKeyList)
GSettingsKeyListValueSet (inGSettings, inValue)	Set value in GSettings by the key list
OSCMD (inCMDStr[, inRunAsyncBool, inLogger])	OS send command in shell locally
OSCredentialsVerify (inUserStr, inPasswordStr)	Verify user credentials in windows.
OSRemotePCRestart (inLogger, inHostStr[, ...])	Send signal via power shell to restart remote PC ATTENTION: Orchestrator user need to have restart right on the Remote machine to restart PC.
OrchestratorIsAdmin ()	Check if Orchestrator process is running as administrator
OrchestratorRerunAsAdmin ()	Check if not admin - then rerun orchestrator as administrator
OrchestratorRestart ([inGSettings])	Orchestrator restart
OrchestratorSessionRestore (inGSettings)	Check _SessionLast_RDPList.json and _SessionLast_StorageDict.pickle in working directory. if exist - load into gsettings # _SessionLast_StorageDict.pickle (binary) _SessionLast_RDPList.json (encoding = “utf-8”) _SessionLast_StorageDict.pickle (binary).
OrchestratorSessionSave (inGSettings)	Orchestrator session save in file
ProcessDefIntervalCall (inGSettings, inDef, ...)	Use this procedure if you need to run periodically some def.

ProcessIsStarted (inProcessNameWOExeStr)	Check if there is any running process that contains the given name processName.
ProcessListGet ([inProcessNameWOExeList])	Return process list on the orchestrator machine sorted by Memory Usage.
ProcessStart (inPathStr, inArgList[, ...])	Start process locally.
ProcessStop (inProcessNameWOExeStr, ...[, ...])	Stop process on the orchestrator machine.
ProcessorActivityItemAppend (inGSettings[, ...])	Create and add activity item in processor queue.
ProcessorActivityItemCreate (inDef[, ...])	Create activity item.
ProcessorAliasDefCreate (inGSettings, inDef)	Create alias for def (can be used in ActivityItem in field Def) !WHEN DEF ALIAS IS REQUIRED! - Def alias is required when you try to call Python def from the Orchestrator WEB side (because you can't transmit Python def object out of the Python environment)
ProcessorAliasDefUpdate (inGSettings, inDef, ...)	Update alias for def (can be used in ActivityItem in field Def).
PythonStart (inModulePathStr, inDefNameStr[, ...])	Import module and run def in the Orchestrator process.
RDPSessionCMDRun (inGSettings, ...[, inModeStr])	Send CMD command to the RDP session "RUN" window
RDPSessionConnect (inGSettings, ...[, ...])	Create new RDPSession in RobotRDPActive. Attention - activity will be ignored if RDP key is already exists
RDPSessionDisconnect (inGSettings, ...[, ...])	Disconnect the RDP session and stop monitoring it.
RDPSessionDublicatesResolve (inGSettings)	DEVELOPING Search duplicates in GSettings RDPlist !def is developing!
RDPSessionFileStoredRecieve (inGSettings, ...)	Recieve file from RDP session to the Orchestrator session using shared drive in RDP (see RDP Configuration Dict, Shared drive)
RDPSessionFileStoredSend (inGSettings, ...)	Send file from Orchestrator session to the RDP session using shared drive in RDP (see RDP Configuration Dict, Shared drive)
RDPSessionLogoff (inGSettings, inRDPSessionKeyStr)	Logoff the RDP session from the Orchestrator process (close all apps in session when logoff)
RDPSessionMonitorStop (inGSettings, ...)	Stop monitoring the RDP session by the Orchestrator process.
RDPSessionProcessStartIfNotRunning (...[, ...])	Start process in RDP if it is not running (check by the arg inProcessNameWEXEStr)
RDPSessionProcessStop (inGSettings, ...)	Send CMD command to the RDP session "RUN" window.
RDPSessionReconnect (inGSettings, ...[, ...])	Reconnect the RDP session
RDPSessionResponsibilityCheck (inGSettings, ...)	DEVELOPING, MAYBE NOT USEFUL Check RDP Session responsibility TODO NEED DEV + TEST
RDPTemplateCreate (inLoginStr, inPasswordStr)	Create RDP connect dict item/ Use it connect/reconnect (Orchestrator.RDPSessionConnect)
SchedulerActivityTimeAddWeekly (inGSettings)	Add activity item list in scheduler.
UACKeyListCheck (inRequest, inRoleKeyList)	Check is client is has access for the key list
UACSuperTokenUpdate (inGSettings, inSuperTokenStr)	Add supertoken for the all access (it is need for the robot communication without human)
UACUpdate (inGSettings, inADLoginStr[, ...])	Update user access (UAC)
UACUserDictGet (inRequest)	Return user UAC hierarchy dict of the inRequest object.
WebCPUpdate (inGSettings, inCPKeyStr[, ...])	Add control panel HTML, JSON generator or JS when page init

WebURLConnectDef (inGSettings, inMethodStr, ...)	Connect URL to DEF
WebURLConnectFile (inGSettings, inMethodStr, ...)	Connect URL to File
WebURLConnectFolder (inGSettings, ...)	Connect URL to Folder
WebUserInfoGet (inRequest)	Return User info about request
WebUserIsSuperToken (inRequest, inGSettings)	Return bool if request is authentificated with supetoken (token which is never expires)
WebUserUACHierarchyGet (inRequest)	Return User UAC Hierarchy DICT Return {...}

pyOpenRPA.Orchestrator.__Orchestrator__.AgentActivityItemAdd(*inGSettings, inHostNameStr, inUserStr, inActivityItemDict*) [source]

Add activity in AgentDict. Check if item is created

Parameters:

- **inGSettings** – Global settings dict (singleton)
- **inHostNameStr** – Agent host name
- **inUserStr** – User login, where agent is based
- **inActivityItemDict** – ActivityItem

Returns:

GUID String of the ActivityItem - you can wait (sync or async) result by this guid!

pyOpenRPA.Orchestrator.__Orchestrator__.AgentActivityItemReturnExists(*inGSettings, inGUIDStr*) [source]

Check by GUID if ActivityItem has been executed and result has come to the Orchestrator

Parameters:

- **inGSettings** – Global settings dict (singleton)
- **inGUIDStr** – GUID String of the ActivityItem - you can wait (sync or async) result by this guid!

Returns:

True - result has been received from the Agent to orc; False - else case

pyOpenRPA.Orchestrator.__Orchestrator__.AgentActivityItemReturnGet(*inGSettings, inGUIDStr, inCheckIntervalSecFloat=0.5*) [source]

Work synchroniously! Wait while result will be recieved. Get the result of the ActivityItem execution on the Agent side. Before this please check by the def AgentActivityItemReturnExists that result has come to the Orchestrator

!ATTENTION! Use only after Orchestrator initialization! Before orchestrator init exception will be raised.

Parameters:

- **inGSettings** – Global settings dict (singleton)
- **inGUIDStr** – GUID String of the ActivityItem - you can wait (sync or async) result by this guid!
- **inCheckIntervalSecFloat** – Interval in sec of the check Activity Item result

Returns:

Result of the ActivityItem executed on the Agent side anr transmitted to the Orchestrator.
IMPORTANT! ONLY JSON ENABLED Types CAN BE TRANSMITTED TO ORCHESTRATOR!

pyOpenRPA.Orchestrator.__Orchestrator__.AgentOSCMD(*inGSettings, inHostNameStr, inUserStr, inCMDStr, inRunAsyncBool=True, inSendOutputToOrchestratorLogsBool=True, inCMDEncodingStr='cp1251'*) [source]

Send CMD to OS thought the pyOpenRPA.Agent daemon. Result return to log + Orchestrator by the

A2O connection

Parameters:

- **inGSettings** – Global settings dict (singleton)
- **inHostNameStr** – Agent host name in upper case (example “RPA01”, “RPA_99” and so on). Active agent session you can see on the orchestrator dashboard as Orchestrator admin
- **inUserStr** – Agent user name in upper case (example “UserRPA”). Active agent session you can see on the orchestrator dashboard as Orchestrator admin
- **inCMDStr** – command to execute on the Agent session
- **inRunAsyncBool** – True - Agent processor don’t wait execution; False - Agent processor wait cmd execution
- **inSendOutputToOrchestratorLogsBool** – True - catch cmd execution output and send it to the Orchestrator logs; Flase - else case; Default True
- **inCMDEncodingStr** – Set the encoding of the DOS window on the Agent server session. Windows is beautiful :) . Default is “cp1251” early was “cp866” - need test

Returns:

GUID String of the ActivityItem - you can wait (sync or async) result by this guid!

---

pyOpenRPA.Orchestrator.__Orchestrator__.AgentOSFileBinaryDataBase64StrCreate(*inGSettings, inHostNameStr, inUserStr, inFilePathStr, inFileDataBase64Str*) [source]

Create binary file by the base64 string by the pyOpenRPA.Agent daemon process (safe for JSON transmission)

Parameters:

- **inGSettings** – Global settings dict (singleton)
- **inHostNameStr** –
- **inUserStr** –
- **inFilePathStr** –
- **inFileDataBase64Str** –

Returns:

GUID String of the ActivityItem - you can wait (sync or async) result by this guid!

---

pyOpenRPA.Orchestrator.__Orchestrator__.AgentOSFileBinaryDataBase64StrReceive(*inGSettings, inHostNameStr, inUserStr, inFilePathStr*) [source]

Read binary file and encode in base64 to transmit (safe for JSON transmition)

Parameters:

- **inGSettings** – Global settings dict (singleton)
- **inHostNameStr** –
- **inUserStr** –
- **inFilePathStr** – File path to read

Returns:

GUID String of the ActivityItem - you can wait (sync or async) result by this guid!

---

pyOpenRPA.Orchestrator.__Orchestrator__.AgentOSFileBinaryDataBytesCreate(*inGSettings, inHostNameStr, inUserStr, inFilePathStr, inFileDataBytes*) [source]

Create binary file by the base64 string by the pyOpenRPA.Agent daemon process (safe for JSON transmition)

Parameters:

- **inGSettings** – Global settings dict (singleton)
- **inHostNameStr** –

- **inUserStr** –
- **inFilePathStr** –
- **inFileDataBytes** –

Returns:
GUID String of the ActivityItem - you can wait (sync or async) result by this guid!

pyOpenRPA.Orchestrator.__Orchestrator__.AgentOSFileTextDataStrCreate(*inGSettings*, *inHostNameStr*, *inUserStr*, *inFilePathStr*, *inFileDataStr*, *inEncodingStr='utf-8'*) [source]

Create text file by the string by the pyOpenRPA.Agent daemon process

Parameters:
- **inGSettings** – Global settings dict (singleton)
- **inHostNameStr** –
- **inUserStr** –
- **inFilePathStr** –
- **inFileDataStr** –
- **inEncodingStr** –

Returns:
GUID String of the ActivityItem - you can wait (sync or async) result by this guid!

pyOpenRPA.Orchestrator.__Orchestrator__.AgentOSFileTextDataStrReceive(*inGSettings*, *inHostNameStr*, *inUserStr*, *inFilePathStr*, *inEncodingStr='utf-8'*) [source]

Read text file in the agent GUI session

Parameters:
- **inGSettings** – Global settings dict (singleton)
- **inHostNameStr** –
- **inUserStr** –
- **inFilePathStr** – File path to read
- **inEncodingStr** – Text file encoding. Default 'utf-8'

Returns:
GUID String of the ActivityItem - you can wait (sync or async) result by this guid!

pyOpenRPA.Orchestrator.__Orchestrator__.AgentProcessWOExeUpperUserListGet(*inGSettings*, *inHostNameStr*, *inUserStr*) [source]

Return the process list only for the current user (where Agent is running) without .EXE in upper case. Can use in ActivityItem from Orchestrator to Agent

Parameters:
- **inGSettings** – Global settings dict (singleton)
- **inHostNameStr** –
- **inUserStr** –

Returns:
GUID String of the ActivityItem - you can wait (sync or async) result by this guid!

pyOpenRPA.Orchestrator.__Orchestrator__.GSettingsAutocleaner(*inGSettings*) [source]

HIDDEN Interval gSettings auto cleaner def to clear some garbage.

Parameters:
inGSettings – Global settings dict (singleton)

Returns:

None

---

pyOpenRPA.Orchestrator.__Orchestrator__.GSettingsKeyListValueAppend(*inGSettings*, *inValue*, *inKeyList=None*) [source]

Append value in GSettings by the key list

```
# USAGE
from pyOpenRPA import Orchestrator

Orchestrator.GSettingsKeyListValueAppend(
   inGSettings = gSettings,
   inValue = "NewValue",
   inKeyList=["NewKeyDict","NewKeyList"]):
# result inGSettings: {
#    ... another keys in gSettings ...,
#    "NewKeyDict":{
#       "NewKeyList":[
#          "NewValue"
#       ]
#    }
#}
```

Parameters:

- **inGSettings** – Global settings dict (singleton)
- **inValue** – Any value to be appended in gSettings Dict by the key list
- **inKeyList** – List of the nested keys (see example)

Returns:

True every time

---

pyOpenRPA.Orchestrator.__Orchestrator__.GSettingsKeyListValueGet(*inGSettings*, *inKeyList=None*) [source]

Get the value from the GSettings by the key list

Parameters:

- **inGSettings** – Global settings dict (singleton)
- **inKeyList** –

Returns:

value any type

---

pyOpenRPA.Orchestrator.__Orchestrator__.GSettingsKeyListValueOperatorPlus(*inGSettings*, *inValue*, *inKeyList=None*) [source]

Execute plus operation between 2 lists (1:inValue and 2:gSettings by the inKeyList)

```
# USAGE
from pyOpenRPA import Orchestrator

Orchestrator.GSettingsKeyListValueOperatorPlus(
   inGSettings = gSettings,
   inValue = [1,2,3],
   inKeyList=["NewKeyDict","NewKeyList"]):
# result inGSettings: {
#    ... another keys in gSettings ...,
#    "NewKeyDict":{
#       "NewKeyList":[
#          "NewValue",
#          1,
#          2,
#          3
#       ]
#    }
#}
```

Parameters:

- **inGSettings** – Global settings dict (singleton)
- **inValue** – List with values to be merged with list in gSettings
- **inKeyList** – List of the nested keys (see example)

Returns:

True every time

---

pyOpenRPA.Orchestrator.__Orchestrator__.GSettingsKeyListValueSet(*inGSettings, inValue, inKeyList=None*) [source]

Set value in GSettings by the key list

Parameters:

- **inGSettings** – Global settings dict (singleton)
- **inValue** –
- **inKeyList** –

Returns:

bool

---

pyOpenRPA.Orchestrator.__Orchestrator__.OSCMD(*inCMDStr, inRunAsyncBool=True, inLogger=None*) [source]

OS send command in shell locally

Parameters:

- **inCMDStr** –
- **inRunAsyncBool** –
- **inLogger** –

Returns:

CMD result string

---

pyOpenRPA.Orchestrator.__Orchestrator__.OSCredentialsVerify(*inUserStr, inPasswordStr, inDomainStr=''*) [source]

Verify user credentials in windows. Return bool

Parameters:

- **inUserStr** –
- **inPasswordStr** –
- **inDomainStr** –

Returns:
True - Credentials are actual; False - Credentials are not actual

---

pyOpenRPA.Orchestrator.__Orchestrator__.OSRemotePCRestart(*inLogger, inHostStr, inForceBool=True*) [source]

Send signal via power shell to restart remote PC ATTENTION: Orchestrator user need to have restart right on the Remote machine to restart PC.

Parameters:
- **inLogger** – logger to log powershell result in logs
- **inHostStr** – PC hostname which you need to restart.
- **inForceBool** – True - send signal to force retart PC; False - else case

Returns:

---

pyOpenRPA.Orchestrator.__Orchestrator__.OrchestratorIsAdmin() [source]

Check if Orchestrator process is running as administrator

Returns:
True - run as administrator; False - not as administrator

---

pyOpenRPA.Orchestrator.__Orchestrator__.OrchestratorRerunAsAdmin() [source]

Check if not admin - then rerun orchestrator as administrator

Returns:
True - run as administrator; False - not as administrator

---

pyOpenRPA.Orchestrator.__Orchestrator__.OrchestratorRestart(*inGSettings=None*) [source]

Orchestrator restart

Parameters:
inGSettings – Global settings dict (singleton)

---

pyOpenRPA.Orchestrator.__Orchestrator__.OrchestratorSessionRestore(*inGSettings*) [source]

Check _SessionLast_RDPList.json and _SessionLast_StorageDict.pickle in working directory. if exist - load into gsettings # _SessionLast_StorageDict.pickle (binary)

_SessionLast_RDPList.json (encoding = “utf-8”) _SessionLast_StorageDict.pickle (binary)

Parameters:
inGSettings – Global settings dict (singleton)

Returns:

---

pyOpenRPA.Orchestrator.__Orchestrator__.OrchestratorSessionSave(*inGSettings*) [source]

Orchestrator session save in file

_SessionLast_RDPList.json (encoding = “utf-8”) _SessionLast_StorageDict.pickle (binary)

Parameters:
inGSettings – Global settings dict (singleton)

Returns:
True every time

---

pyOpenRPA.Orchestrator.__Orchestrator__.ProcessDefIntervalCall(*inGSettings, inDef, inIntervalSecFloat, inIntervalAsyncBool=False, inDefArgList=None, inDefArgDict=None, inDefArgGSettingsNameStr=None, inDefArgLoggerNameStr=None, inExecuteInNewThreadBool=True, inLogger=None*) [source]

Use this procedure if you need to run periodically some def. Set def, args, interval and enjoy :)

Parameters:

- **inGSettings** – global settings
- **inDef** – def link, which will be called with interval inIntervalSecFloat
- **inIntervalSecFloat** – Interval in seconds between call
- **inIntervalAsyncBool** – False - wait interval before next call after the previous iteration result; True - wait interval after previous iteration call
- **inDefArgList** – List of the args in def. Default None (empty list)
- **inDefArgDict** – Dict of the args in def. Default None (empty dict)
- **inDefArgGSettingsNameStr** – Name of the GSettings arg name for def (optional)
- **inDefArgLoggerNameStr** – Name of the Logger arg name for def (optional). If Use - please check fill of the inLogger arg.
- **inExecuteInNewThreadBool** – True - create new thread for the periodic execution; False - execute in current thread. Default: True
- **inLogger** – logging def if some case is appear

Returns:

pyOpenRPA.Orchestrator.__Orchestrator__.ProcessIsStarted(*inProcessNameWOExeStr*) [source]

Check if there is any running process that contains the given name processName.

```
# USAGE
from pyOpenRPA import Orchestrator

IProcessIsStartedBool = Orchestrator.ProcessIsStarted(inProcessNameWOExeStr = "notepad")
# IProcessIsStartedBool is True - notepad.exe is running on the Orchestrator machine
```

Parameters:

inProcessNameWOExeStr – Process name WithOut (WO) '.exe' postfix. Example: "notepad" (not "notepad.exe")

Returns:

True - process is running on the orchestrator machine; False - process is not running on the orchestrator machine

pyOpenRPA.Orchestrator.__Orchestrator__.ProcessListGet(*inProcessNameWOExeList=None*) [source]

Return process list on the orchestrator machine sorted by Memory Usage. You can determine the list of the processes you are interested - def will return the list about it.

```
# USAGE
from pyOpenRPA import Orchestrator

IProcessList = Orchestrator.ProcessListGet()
# Return the list of the process on the machine.
# !ATTENTION! RUn orchestrator as administrator to get all process list on the machine.
```

Parameters:

inProcessNameWOExeList –

Returns:

{

"ProcessWOExeList": ["notepad","..."], "ProcessWOExeUpperList": ["NOTEPAD","..."],

“ProcessDetailList”: [

{

‘pid’: 412, ‘username’: “DESKTOPUSER”, ‘name’: ‘notepad.exe’, ‘vms’: 13.77767775, ‘NameWOExeUpperStr’: ‘NOTEPAD’, ‘NameWOExeStr’: “‘notepad’”},

{…}]

pyOpenRPA.Orchestrator.__Orchestrator__.ProcessStart(*inPathStr, inArgList, inStopProcessNameWOExeStr=None*) [source]

Start process locally. Extra feature: Use inStopProcessNameWOExeStr to stop the execution if current process is running.

```
# USAGE
from pyOpenRPA import Orchestrator

Orchestrator.ProcessStart(
   inPathStr = "notepad"
   inArgList = []
   inStopProcessNameWOExeStr = "notepad")
# notepad.exe will be started if no notepad.exe is active on the machine
```

Parameters:

- **inPathStr** – Command to send in CMD
- **inArgList** – List of the arguments for the CMD command. Example: [“test.txt”]
- **inStopProcessNameWOExeStr** – Trigger: stop execution if process is running. Process name WithOut (WO) ‘.exe’ postfix. Example: “notepad” (not “notepad.exe”)

Returns:

None - nothing is returned. If process will not start -exception will be raised

pyOpenRPA.Orchestrator.__Orchestrator__.ProcessStop(*inProcessNameWOExeStr, inCloseForceBool, inUserNameStr='%username%'*) [source]

Stop process on the orchestrator machine. You can set user session on the machine and set flag about to force close process.

```
# USAGE
from pyOpenRPA import Orchestrator

Orchestrator.ProcessStop(
   inProcessNameWOExeStr = "notepad"
   inCloseForceBool = True
   inUserNameStr = "USER_99")
# Will close process "notepad.exe" on the user session "USER_99" (!ATTENTION! if process not exists no exceptions will be raised)
```

Parameters:

- **inProcessNameWOExeStr** – Process name WithOut (WO) ‘.exe’ postfix. Example: “notepad” (not “notepad.exe”)
- **inCloseForceBool** – True - do force close. False - send signal to safe close (!ATTENTION! - Safe close works only in orchestrator session. Win OS doens’t allow to send safe close signal between GUI sessions)
- **inUserNameStr** – User name which is has current process to close. Default value is close process on the Orchestrator session

Returns:

None

pyOpenRPA.Orchestrator.__Orchestrator__.ProcessorActivityItemAppend(*inGSettings, inDef=None, inArgList=None, inArgDict=None, inArgGSettingsStr=None, inArgLoggerStr=None, inActivityItemDict=None*) [source]

Create and add activity item in processor queue.

```
# USAGE
from pyOpenRPA import Orchestrator

# EXAMPLE 1
def TestDef(inArg1Str, inGSettings, inLogger):
    pass
IActivityItem = Orchestrator.ProcessorActivityItemAppend(
    inGSettings = gSettingsDict,
    inDef = TestDef,
    inArgList=[],
    inArgDict={"inArg1Str": "ArgValueStr"},
    inArgGSettingsStr = "inGSettings",
    inArgLoggerStr = "inLogger")
# Activity have been already append in the processor queue

# EXAMPLE 2
def TestDef(inArg1Str):
    pass
Orchestrator.ProcessorAliasDefUpdate(
    inGSettings = gSettings,
    inDef = TestDef,
    inAliasStr="TestDefAlias")
IActivityItem = Orchestrator.ProcessorActivityItemCreate(
    inDef = "TestDefAlias",
    inArgList=[],
    inArgDict={"inArg1Str": "ArgValueStr"},
    inArgGSettingsStr = None,
    inArgLoggerStr = None)
Orchestrator.ProcessorActivityItemAppend(
    inGSettings = gSettingsDict,
    inActivityItemDict = IActivityItem)
# Activity have been already append in the processor queue
```

Parameters:

- **inGSettings** – Global settings dict (singleton)
- **inDef** – def link or def alias (look gSettings[“Processor”][“AliasDefDict”])
- **inArgList** – Args list for the Def
- **inArgDict** – Args dict for the Def
- **inArgGSettingsStr** – Name of def argument of the GSettings dict
- **inArgLoggerStr** – Name of def argument of the logging object
- **inActivityItemDict** – Fill if you already have ActivityItemDict (don’t fill inDef, inArgList, inArgDict, inArgGSettingsStr, inArgLoggerStr)

:return ActivityItem GUIDStr

---

pyOpenRPA.Orchestrator.__Orchestrator__.ProcessorActivityItemCreate(*inDef, inArgList=None, inArgDict=None, inArgGSettingsStr=None, inArgLoggerStr=None, inGUIDStr=None*) [source]

Create activity item. Activity item can be used as list item in ProcessorActivityItemAppend or in Processor.ActivityListExecute.

```
# USAGE
from pyOpenRPA import Orchestrator

# EXAMPLE 1
def TestDef(inArg1Str, inGSettings, inLogger):
    pass
IActivityItem = Orchestrator.ProcessorActivityItemCreate(
    inDef = TestDef,
    inArgList=[],
    inArgDict={"inArg1Str": "ArgValueStr"},
    inArgGSettingsStr = "inGSettings",
    inArgLoggerStr = "inLogger")
# IActivityItem:
#  {
#      "Def":TestDef,
#      "ArgList":inArgList,
#      "ArgDict":inArgDict,
#      "ArgGSettings": "inArgGSettings",
#      "ArgLogger": "inLogger"
#  }

# EXAMPLE 2
def TestDef(inArg1Str):
    pass
Orchestrator.ProcessorAliasDefUpdate(
    inGSettings = gSettings,
    inDef = TestDef,
    inAliasStr="TestDefAlias")
IActivityItem = Orchestrator.ProcessorActivityItemCreate(
    inDef = "TestDefAlias",
    inArgList=[],
    inArgDict={"inArg1Str": "ArgValueStr"},
    inArgGSettingsStr = None,
    inArgLoggerStr = None)
# IActivityItem:
#  {
#      "Def":"TestDefAlias",
#      "ArgList":inArgList,
#      "ArgDict":inArgDict,
#      "ArgGSettings": None,
#      "ArgLogger": None
#  }
```

Parameters:

- **inDef** – def link or def alias (look gSettings[“Processor”][“AliasDefDict”])
- **inArgList** – Args list for the Def
- **inArgDict** – Args dict for the def
- **inArgGSettingsStr** – Name of def argument of the GSettings dict
- **inArgLoggerStr** – Name of def argument of the logging object
- **inGUIDStr** – GUID which you can specify. If None the GUID will be generated

Returns:

{}

pyOpenRPA.Orchestrator.__Orchestrator__.ProcessorAliasDefCreate(*inGSettings*, *inDef*, *inAliasStr=None*)
[source]

Create alias for def (can be used in ActivityItem in field Def) !WHEN DEF ALIAS IS REQUIRED! - Def alias is required when you try to call Python def from the Orchestrator WEB side (because you can’t transmit Python def object out of the Python environment)

```
# USAGE
from pyOpenRPA import Orchestrator

def TestDef():
    pass
lAliasStr = Orchestrator.ProcessorAliasDefCreate(
    inGSettings = gSettings,
    inDef = TestDef,
    inAliasStr="TestDefAlias")
# Now you can call TestDef by the alias from var lAliasStr with help of ActivityItem (key Def = lAliasStr)
```

Parameters:

- **inGSettings** – Global settings dict (singleton)
- **inDef** – Def
- **inAliasStr** – String alias for associated def

Returns:

str Alias string (Alias can be regenerated if previous alias was occupied)

---

pyOpenRPA.Orchestrator.__Orchestrator__.ProcessorAliasDefUpdate(*inGSettings, inDef, inAliasStr*) [source]

Update alias for def (can be used in ActivityItem in field Def). !WHEN DEF ALIAS IS REQUIRED! - Def alias is required when you try to call Python def from the Orchestrator WEB side (because you can't transmit Python def object out of the Python environment)

```
# USAGE
from pyOpenRPA import Orchestrator

def TestDef():
    pass
Orchestrator.ProcessorAliasDefUpdate(
    inGSettings = gSettings,
    inDef = TestDef,
    inAliasStr="TestDefAlias")
# Now you can call TestDef by the alias "TestDefAlias" with help of ActivityItem (key Def = "TestDefAlias")
```

Parameters:

- **inGSettings** – Global settings dict (singleton)
- **inDef** – Def
- **inAliasStr** – String alias for associated def

Returns:

str Alias string

---

pyOpenRPA.Orchestrator.__Orchestrator__.PythonStart(*inModulePathStr, inDefNameStr, inArgList=None, inArgDict=None, inLogger=None*) [source]

Import module and run def in the Orchestrator process.

Note

Import module will be each time when PythonStart def will be called.

```
# USAGE
from pyOpenRPA import Orchestrator

Orchestrator.PythonStart(
    inModulePathStr="ModuleToCall.py", # inModulePathStr: Working Directory\ModuleToCall.py
    inDefNameStr="TestDef")
# Import module in Orchestrator process and call def "TestDef" from module "ModuleToCall.py"
```

Parameters:

- **inModulePathStr** – Absolute or relative (working directory of the orchestrator process) path to the importing module .py
- **inDefNameStr** – Def name in module
- **inArgList** – List of the arguments for callable def
- **inArgDict** – Dict of the named arguments for callable def
- **inLogger** – Logger instance to log some information when PythonStart def is running

Returns:

None

---

pyOpenRPA.Orchestrator.__Orchestrator__.RDPSessionCMDRun(*inGSettings, inRDPSessionKeyStr, inCMDStr, inModeStr='CROSSCHECK'*) [source]

Send CMD command to the RDP session “RUN” window

```
# USAGE
from pyOpenRPA import Orchestrator

IResultDict = Orchestrator.RDPSessionCMDRun(
    inGSettings = gSettings,
    inRDPSessionKeyStr = "RDPKey",
    inModeStr = 'LISTEN')
# Orchestrator will send CMD to RDP and return the result (see return section)
```

Parameters:

- **inGSettings** – Global settings dict (singleton)
- **inRDPSessionKeyStr** – RDP Session string key - need for the further identification
- **inCMDStr** – Any CMD string
- **inModeStr** – Variants: “LISTEN” - Get result of the cmd command in result; “CROSSCHECK” - Check if the command was successfully sent “RUN” - Run without crosscheck and get clipboard

Returns:

 # OLD > True - CMD was executed successfully

{

“OutStr”: <> # Result string “IsResponsibleBool”: True|False # Flag is RDP is responsible - works only when inModeStr = CROSSCHECK

}

---

pyOpenRPA.Orchestrator.__Orchestrator__.RDPSessionConnect(*inGSettings, inRDPSessionKeyStr, inRDPTemplateDict=None, inHostStr=None, inPortStr=None, inLoginStr=None, inPasswordStr=None*) [source]

Create new RDPSession in RobotRDPActive. Attention - activity will be ignored if RDP key is already exists

2 way of the use

Var 1 (Main stream): inGSettings, inRDPSessionKeyStr, inRDPTemplateDict Var 2 (Backward compatibility): inGSettings, inRDPSessionKeyStr, inHostStr, inPortStr, inLoginStr, inPasswordStr

```
# USAGE
from pyOpenRPA import Orchestrator

IRDPItemDict = Orchestrator.RDPTemplateCreate(
   inLoginStr = "USER_99",
   inPasswordStr = "USER_PASS_HERE", inHostStr="127.0.0.1", inPortInt = 3389, inWidthPXInt = 1680,
   inHeightPXInt = 1050, inUseBothMonitorBool = False, inDepthBitInt = 32, inSharedDriveList=None)
Orchestrator.RDPSessionConnect(
   inGSettings = gSettings,
   inRDPSessionKeyStr = "RDPKey",
   inRDPTemplateDict = IRDPItemDict)
# Orchestrator will create RDP session by the IRDPItemDict configuration
```

Parameters:

- **inGSettings** – Global settings dict (singleton)
- **inRDPSessionKeyStr** – RDP Session string key - need for the further identification
- **inRDPTemplateDict** – RDP configuration dict with settings (see def Orchestrator.RDPTemplateCreate)
- **inHostStr** – Backward compatibility from Orchestrator v 1.1.20. Use inRDPTemplateDict
- **inPortStr** – Backward compatibility from Orchestrator v 1.1.20. Use inRDPTemplateDict
- **inLoginStr** – Backward compatibility from Orchestrator v 1.1.20. Use inRDPTemplateDict
- **inPasswordStr** – Backward compatibility from Orchestrator v 1.1.20. Use inRDPTemplateDict

Returns:

True every time :)

---

pyOpenRPA.Orchestrator.__Orchestrator__.RDPSessionDisconnect(*inGSettings, inRDPSessionKeyStr, inBreakTriggerProcessWOExeList=None*) [source]

Disconnect the RDP session and stop monitoring it.

```
# USAGE
from pyOpenRPA import Orchestrator

Orchestrator.RDPSessionDisconnect(
   inGSettings = gSettings,
   inRDPSessionKeyStr = "RDPKey")
# Orchestrator will disconnect RDP session and will stop to monitoring current RDP
```

Parameters:

- **inGSettings** – Global settings dict (singleton)
- **inRDPSessionKeyStr** – RDP Session string key - need for the further identification
- **inBreakTriggerProcessWOExeList** –
  List of the processes, which will stop the execution. Example [“notepad”]
  Orchestrator look processes on the current machine

Returns:

True every time

---

pyOpenRPA.Orchestrator.__Orchestrator__.RDPSessionDublicatesResolve(*inGSettings*) [source]

DEVELOPING Search duplicates in GSettings RDPlist !def is developing!

Parameters:

inGSettings – Global settings dict (singleton)

Returns:

---

pyOpenRPA.Orchestrator.__Orchestrator__.RDPSessionFileStoredRecieve(*inGSettings, inRDPSessionKeyStr, inRDPFilePathStr, inHostFilePathStr*) [source]

Recieve file from RDP session to the Orchestrator session using shared drive in RDP (see RDP

Configuration Dict, Shared drive)

```
# USAGE
from pyOpenRPA import Orchestrator

lResultDict = Orchestrator.RDPSessionFileStoredRecieve(
   inGSettings = gSettings,
   inRDPSessionKeyStr = "RDPKey",
   inHostFilePathStr = "TESTDIR\Test.py",
   inRDPFilePathStr = "C:\RPA\TESTDIR\Test.py")
# Orchestrator will send CMD to RDP and return the result (see return section)
```

Parameters:

- **inGSettings** – Global settings dict (singleton)
- **inRDPSessionKeyStr** – RDP Session string key - need for the further identification
- **inRDPFilePathStr** – !Absolute! path to the destination file location on the RDP side. Example: “C:RPATESTDIRTest.py”
- **inHostFilePathStr** – Relative or absolute path to the file location on the Orchestrator side. Example: “TESTDIRTest.py”

Returns:

True every time

---

pyOpenRPA.Orchestrator.__Orchestrator__.RDPSessionFileStoredSend(*inGSettings, inRDPSessionKeyStr, inHostFilePathStr, inRDPFilePathStr*) [source]

Send file from Orchestrator session to the RDP session using shared drive in RDP (see RDP Configuration Dict, Shared drive)

```
# USAGE
from pyOpenRPA import Orchestrator

lResultDict = Orchestrator.RDPSessionFileStoredSend(
   inGSettings = gSettings,
   inRDPSessionKeyStr = "RDPKey",
   inHostFilePathStr = "TESTDIR\Test.py",
   inRDPFilePathStr = "C:\RPA\TESTDIR\Test.py")
# Orchestrator will send CMD to RDP and return the result (see return section)
```

Parameters:

- **inGSettings** – Global settings dict (singleton)
- **inRDPSessionKeyStr** – RDP Session string key - need for the further identification
- **inHostFilePathStr** – Relative or absolute path to the file location on the Orchestrator side. Example: “TESTDIRTest.py”
- **inRDPFilePathStr** – !Absolute! path to the destination file location on the RDP side. Example: “C:RPATESTDIRTest.py”

Returns:

True every time

---

pyOpenRPA.Orchestrator.__Orchestrator__.RDPSessionLogoff(*inGSettings, inRDPSessionKeyStr, inBreakTriggerProcessWOExeList=None*) [source]

Logoff the RDP session from the Orchestrator process (close all apps in session when logoff)

```
# USAGE
from pyOpenRPA import Orchestrator

Orchestrator.RDPSessionLogoff(
   inGSettings = gSettings,
   inRDPSessionKeyStr = "RDPKey",
   inBreakTriggerProcessWOExeList = ['Notepad'])
# Orchestrator will logoff the RDP session
```

Parameters:

- **inGSettings** – Global settings dict (singleton)
- **inRDPSessionKeyStr** – RDP Session string key - need for the further identification
- **inBreakTriggerProcessWOExeList** – List of the processes, which will stop the execution. Example [“notepad”]

Returns:

True - logoff is successful

---

pyOpenRPA.Orchestrator.__Orchestrator__.RDPSessionMonitorStop(*inGSettings, inRDPSessionKeyStr*) [source]

Stop monitoring the RDP session by the Orchestrator process. Current def don’t kill RDP session - only stop to track it (it can give)

```
# USAGE
from pyOpenRPA import Orchestrator

Orchestrator.RDPSessionMonitorStop(
   inGSettings = gSettings,
   inRDPSessionKeyStr = "RDPKey")
# Orchestrator will stop the RDP monitoring
```

Parameters:

- **inGSettings** – Global settings dict (singleton)
- **inRDPSessionKeyStr** – RDP Session string key - need for the further identification

Returns:

True every time :>

---

pyOpenRPA.Orchestrator.__Orchestrator__.RDPSessionProcessStartIfNotRunning(*inGSettings, inRDPSessionKeyStr, inProcessNameWEXEStr, inFilePathStr, inFlagGetAbsPathBool=True*) [source]

Start process in RDP if it is not running (check by the arg inProcessNameWEXEStr)

```
# USAGE
from pyOpenRPA import Orchestrator

Orchestrator.RDPSessionProcessStartIfNotRunning(
   inGSettings = gSettings,
   inRDPSessionKeyStr = "RDPKey",
   inProcessNameWEXEStr = 'Notepad.exe',
   inFilePathStr = "path\to    he\executable\file.exe"
   inFlagGetAbsPathBool = True)
# Orchestrator will start the process in RDP session
```

Parameters:

- **inGSettings** – Global settings dict (singleton)
- **inRDPSessionKeyStr** – RDP Session string key - need for the further identification
- **inProcessNameWEXEStr** – Process name with extension (.exe). This arg allow to check the process is running. Example: “Notepad.exe”
- **inFilePathStr** – Path to run process if it is not running.
- **inFlagGetAbsPathBool** – True - get abs path from the relative path in inFilePathStr. False - else

case

Returns:

True every time :)

---

pyOpenRPA.Orchestrator.__Orchestrator__.RDPSessionProcessStop(*inGSettings, inRDPSessionKeyStr, inProcessNameWEXEStr, inFlagForceCloseBool*) [source]

Send CMD command to the RDP session “RUN” window.

```
# USAGE
from pyOpenRPA import Orchestrator

IResultDict = Orchestrator.RDPSessionProcessStop(
    inGSettings = gSettings,
    inRDPSessionKeyStr = "RDPKey",
    inProcessNameWEXEStr = 'notepad.exe',
    inFlagForceCloseBool = True)
# Orchestrator will send CMD to RDP and return the result (see return section)
```

Parameters:

- **inGSettings** – Global settings dict (singleton)
- **inRDPSessionKeyStr** – RDP Session string key - need for the further identification
- **inProcessNameWEXEStr** – Process name to kill. Example: ‘notepad.exe’
- **inFlagForceCloseBool** – True - force close the process. False - safe close the process

Returns:

True every time

---

pyOpenRPA.Orchestrator.__Orchestrator__.RDPSessionReconnect(*inGSettings, inRDPSessionKeyStr, inRDPTemplateDict=None*) [source]

Reconnect the RDP session

```
# USAGE
from pyOpenRPA import Orchestrator

IRDPItemDict = Orchestrator.RDPTemplateCreate(
    inLoginStr = "USER_99",
    inPasswordStr = "USER_PASS_HERE", inHostStr="127.0.0.1", inPortInt = 3389, inWidthPXInt = 1680,
    inHeightPXInt = 1050, inUseBothMonitorBool = False, inDepthBitInt = 32, inSharedDriveList=None)
Orchestrator.RDPSessionReconnect(
    inGSettings = gSettings,
    inRDPSessionKeyStr = "RDPKey",
    inRDPTemplateDict = inRDPTemplateDict)
# Orchestrator will reconnect RDP session and will continue to monitoring current RDP
```

Parameters:

- **inGSettings** – Global settings dict (singleton)
- **inRDPSessionKeyStr** – RDP Session string key - need for the further identification
- **inRDPTemplateDict** – RDP configuration dict with settings (see def Orchestrator.RDPTemplateCreate)

Returns:

---

pyOpenRPA.Orchestrator.__Orchestrator__.RDPSessionResponsibilityCheck(*inGSettings, inRDPSessionKeyStr*) [source]

DEVELOPING, MAYBE NOT USEFUL Check RDP Session responsibility TODO NEED DEV + TEST

Parameters:

- **inGSettings** – Global settings dict (singleton)
- **inRDPSessionKeyStr** – RDP Session string key - need for the further identification

Returns:

True every time

---

pyOpenRPA.Orchestrator.__Orchestrator__.RDPTemplateCreate(*inLoginStr, inPasswordStr, inHostStr='127.0.0.1', inPortInt=3389, inWidthPXInt=1680, inHeightPXInt=1050, inUseBothMonitorBool=False, inDepthBitInt=32, inSharedDriveList=None*) [source]

Create RDP connect dict item/ Use it connect/reconnect (Orchestrator.RDPSessionConnect)

```
# USAGE
from pyOpenRPA import Orchestrator

IRDPItemDict = Orchestrator.RDPTemplateCreate(
   inLoginStr = "USER_99",
   inPasswordStr = "USER_PASS_HERE",
   inHostStr="127.0.0.1",
   inPortInt = 3389,
   inWidthPXInt = 1680,
   inHeightPXInt = 1050,
   inUseBothMonitorBool = False,
   inDepthBitInt = 32,
   inSharedDriveList=None)
#    IRDPTemplateDict= { # Init the configuration item
#        "Host": "127.0.0.1", "Port": "3389", "Login": "USER_99", "Password": "USER_PASS_HERE",
#        "Screen": { "Width": 1680, "Height": 1050, "FlagUseAllMonitors": False, "DepthBit": "32" },
#        "SharedDriveList": ["c"],
#        ####### Will updated in program ##############
#        "SessionHex": "77777sdfsdf77777dsfdfsf77777777",  # Hex is created when robot runs, example ""
#        "SessionIsWindowExistBool": False, "SessionIsWindowResponsibleBool": False, "SessionIsIgnoredBool": False
#    }
```

Parameters:

- **inLoginStr** – User/Robot Login, example “USER_99”
- **inPasswordStr** – Password, example “USER_PASS_HERE”
- **inHostStr** – Host address, example “77.77.22.22”
- **inPortInt** – RDP Port, example “3389” (default)
- **inWidthPXInt** – Width of the remote desktop in pixels, example 1680
- **inHeightPXInt** – Height of the remote desktop in pixels, example 1050
- **inUseBothMonitorBool** – True - connect to the RDP with both monitors. False - else case
- **inDepthBitInt** – Remote desktop bitness. Available: 32 or 24 or 16 or 15, example 32
- **inSharedDriveList** – Host local disc to connect to the RDP session. Example: [“c”, “d”]

Returns:

{

“Host”: inHostStr, # Host address, example “77.77.22.22” “Port”: str(inPortInt), # RDP Port, example “3389” “Login”: inLoginStr, # Login, example “test” “Password”: inPasswordStr, # Password, example “test” “Screen”: {

”Width”: inWidthPXInt, # Width of the remote desktop in pixels, example 1680 “Height”: inHeightPXInt, # Height of the remote desktop in pixels, example 1050 # “640x480” or “1680x1050” or “FullScreen”. If Resolution not exists set full screen, example “FlagUseAllMonitors”: inUseBothMonitorBool, # True or False, example False “DepthBit”: str(inDepthBitInt) # “32” or “24” or “16” or “15”, example “32”

}, “SharedDriveList”: inSharedDriveList, # List of the Root sesion hard drives, example [“c”] ###### Will updated in program ############ “SessionHex”: “77777sdfsdf77777dsfdfsf77777777”, # Hex is created when robot runs, example “” “SessionIsWindowExistBool”: False, # Flag if the RDP window is exist, old name “FlagSessionIsActive”. Check every n seconds , example False “SessionIsWindowResponsibleBool”: False, # Flag if RDP window is responsible (recieve

commands). Check every nn seconds. If window is Responsible - window is Exist too , example False “SessionIsIgnoredBool”: False # Flag to ignore RDP window False - dont ignore, True - ignore, example False

}

pyOpenRPA.Orchestrator.__Orchestrator__.SchedulerActivityTimeAddWeekly(*inGSettings, inTimeHHMMStr='23:55:', inWeekdayList=None, inActivityList=None*) [source]

Add activity item list in scheduler. You can set weekday list and set time when launch. Activity list will be executed at planned time/day.

```
# USAGE
from pyOpenRPA import Orchestrator

# EXAMPLE 1
def TestDef(inArg1Str):
    pass
lActivityItem = Orchestrator.ProcessorActivityItemCreate(
    inDef = TestDef,
    inArgList=[],
    inArgDict={"inArg1Str": "ArgValueStr"},
    inArgGSettingsStr = None,
    inArgLoggerStr = None)
Orchestrator.SchedulerActivityTimeAddWeekly(
    inGSettings = gSettingsDict,
    inTimeHHMMStr = "04:34",
    inWeekdayList=[2,3,4],
    inActivityList = [lActivityItem])
# Activity will be executed at 04:34 Wednesday (2), thursday (3), friday (4)
```

Parameters:

- **inGSettings** – Global settings dict (singleton)
- **inTimeHHMMStr** – Activation time from “00:00” to “23:59”. Example: “05:29”
- **inWeekdayList** – Week day list to initiate activity list. Use int from 0 (monday) to 6 (sunday) as list items. Example: [0,1,2,3,4]. Default value is everyday ([0,1,2,3,4,5,6])
- **inActivityList** – Activity list structure

Returns:

None

pyOpenRPA.Orchestrator.__Orchestrator__.UACKeyListCheck(*inRequest, inRoleKeyList*) → bool [source]

Check is client is has access for the key list

Parameters:

- **inRequest** – request handler (from http.server import BaseHTTPRequestHandler)
- **inRoleKeyList** –

Returns:

bool

pyOpenRPA.Orchestrator.__Orchestrator__.UACSuperTokenUpdate(*inGSettings, inSuperTokenStr*) [source]

Add supertoken for the all access (it is need for the robot communication without human)

Parameters:

- **inGSettings** – Global settings dict (singleton)
- **inSuperTokenStr** –

pyOpenRPA.Orchestrator.__Orchestrator__.UACUpdate(*inGSettings, inADLoginStr, inADStr='', inADIsDefaultBool=True, inURLList=None, inRoleHierarchyAllowedDict=None*) [source]

Update user access (UAC)

Parameters:

- **inGSettings** – Global settings dict (singleton)
- **inADLoginStr** –
- **inADStr** –
- **inADIsDefaultBool** –
- **inURLList** –
- **inRoleHierarchyAllowedDict** –

---

pyOpenRPA.Orchestrator.__Orchestrator__.UACUserDictGet(*inRequest*) → dict [source]

Return user UAC hierarchy dict of the inRequest object. Empty dict - superuser access

Parameters:

inRequest – request handler (from http.server import BaseHTTPRequestHandler)

Returns:

user UAC hierarchy dict

---

pyOpenRPA.Orchestrator.__Orchestrator__.WebCPUpdate(*inGSettings*, *inCPKeyStr*, *inHTMLRenderDef=None*, *inJSONGeneratorDef=None*, *inJSInitGeneratorDef=None*) [source]

Add control panel HTML, JSON generator or JS when page init

Parameters:

- **inGSettings** – Global settings dict (singleton)
- **inCPKeyStr** –
- **inHTMLRenderDef** –
- **inJSONGeneratorDef** –
- **inJSInitGeneratorDef** –

---

pyOpenRPA.Orchestrator.__Orchestrator__.WebURLConnectDef(*inGSettings*, *inMethodStr*, *inURLStr*, *inMatchTypeStr*, *inDef*, *inContentTypeStr='application/octet-stream'*) [source]

> **Connect URL to DEF**
>
> "inMethodStr":"GET|POST", "inURLStr": "/index", #URL of the request "inMatchTypeStr": "", #"BeginWith|Contains|Equal|EqualCase", "inContentTypeStr": "", #HTTP Content-type "inDef": None #Function with str result

Parameters:

- **inGSettings** – Global settings dict (singleton)
- **inMethodStr** –
- **inURLStr** –
- **inMatchTypeStr** –
- **inDef** –
- **inContentTypeStr** –

---

pyOpenRPA.Orchestrator.__Orchestrator__.WebURLConnectFile(*inGSettings*, *inMethodStr*, *inURLStr*, *inMatchTypeStr*, *inFilePathStr*, *inContentTypeStr='application/octet-stream'*) [source]

> **Connect URL to File**
>
> "inMethodStr":"GET|POST", "inURLStr": "/index", #URL of the request "inMatchTypeStr": "", #"BeginWith|Contains|Equal|EqualCase", "inFolderPathStr": "", #Absolute or relative path

Parameters:

- **inGSettings** – Global settings dict (singleton)
- **inMethodStr** –

- **inURLStr** –
- **inMatchTypeStr** –
- **inFilePathStr** –
- **inContentTypeStr** –

pyOpenRPA.Orchestrator.__Orchestrator__.WebURLConnectFolder(*inGSettings, inMethodStr, inURLStr, inMatchTypeStr, inFolderPathStr*) [source]

Connect URL to Folder

"inMethodStr":"GET|POST", "inURLStr": "/Folder/", #URL of the request "inMatchTypeStr": "", #"BeginWith|Contains|Equal|EqualCase", "inFolderPathStr": "", #Absolute or relative path

Parameters:

- **inGSettings** – Global settings dict (singleton)
- **inMethodStr** –
- **inURLStr** –
- **inMatchTypeStr** –
- **inFolderPathStr** –

pyOpenRPA.Orchestrator.__Orchestrator__.WebUserInfoGet(*inRequest*) [source]

Return User info about request

Parameters:
inRequest –

Returns:
{"DomainUpperStr": "", "UserNameUpperStr": ""}

pyOpenRPA.Orchestrator.__Orchestrator__.WebUserIsSuperToken(*inRequest, inGSettings*) [source]

Return bool if request is authentificated with supetoken (token which is never expires)

Parameters:

- **inRequest** –
- **inGSettings** – Global settings dict (singleton)

Returns:
bool True - is supertoken; False - is not supertoken

pyOpenRPA.Orchestrator.__Orchestrator__.WebUserUACHierarchyGet(*inRequest*) [source]

Return User UAC Hierarchy DICT Return {...}

Parameters:
inRequest –

Returns:
UAC Dict {}

pyOpenRPA.Orchestrator.Web.Basic

Functions:

JSActivityListExecute(inActivityList[, ...])	Create JS for execute activity list/ activity permanent
JSProcessorActivityListAdd(inActivityList[, ...])	# Create JS for send activity list/ activity to the processor

pyOpenRPA.Orchestrator.Web.Basic.JSActivityListExecute(*inActivityList, inGUIDRemoveBool=True*) [source]

Create JS for execute activity list/ activity permanent USAGE:
Orchestrator.Web.Basic.JSActivityListExecute(inActivityList, inGUIDRemoveBool = True)

Parameters:

- **inActivityList** – List of the activities (See __Orchestrator__.ProcessorActivityItemCreate)
- **inGUIDRemoveBool** – True - remove GUID before generate JS (if GUID is not important)

Returns:

JavaScript string for the front end

pyOpenRPA.Orchestrator.Web.Basic.JSProcessorActivityListAdd(*inActivityList, inGUIDRemoveBool=True*) [source]

# Create JS for send activity list/ activity to the processor # USAGE:
Orchestrator.Web.Basic.JSProcessorActivityListAdd(inActivityList)

Parameters:

- **inActivityList** – List of the activities (See __Orchestrator__.ProcessorActivityItemCreate)
- **inGUIDRemoveBool** – True - remove GUID before generate JS (if GUID is not important)

Returns:

JavaScript string for the front end

References

reStructuredText [1]

[1] :

http://docutils.sourceforge.net/rst.html

Previous Next

3. gSettings Template

gSettings structure

```
import os, logging, datetime, sys

# Technical def - return GSettings structure with examples
def __Create__():
    return {
        "VersionStr": None, # Will be filled in orchestrator,
        "Autocleaner": {
            # Some gurbage is collecting in g settings. So you can configure autocleaner to periodically clear gSettings
            "IntervalSecFloat": 3600.0, # Sec float to periodically clear gsettings
            "AgentActivityReturnLifetimeSecFloat": 300.0 # Time in seconds to life for activity result recieved from the agent
        },
        "Client": { # Settings about client web orchestrator
            "Session": {
                # Settings about web session. Session algorythms works only for special requests (URL in ServerSettings)
                "LifetimeSecFloat": 600.0,
                # Client Session lifetime in seconds. after this time server will forget about this client session
                "LifetimeRequestSecFloat": 120.0, # 1 client request lifetime in server in seconds
                "ControlPanelRefreshIntervalSecFloat": 2.0, # Interval to refresh control panels for session,
                "TechnicalSessionGUIDCache": { # TEchnical cache. Fills when web browser is requesting
                    # "SessionGUIDStr":{ # Session with some GUID str. On client session guid stored in cookie "SessionGUIDStr"
                    #   "InitDatetime": None, # Datetime when session GUID was created
                    #   "DatasetLast": {
                    #       "ControlPanel": {
                    #           "Data": None, # Struct to check with new iterations. None if starts
                    #           "ReturnBool": False # flag to return, close request and return data as json
                    #       }
                    #   },
                    #   "ClientRequestHandler": None, # Last client request handler
                    #   "UserADStr": None, # User, who connect. None if user is not exists
                    #   "DomainADStr": None, # Domain of the user who connect. None if user is not exists
                    # }
                },
            },
            ###### Client... ########
            "DumpLogListRefreshIntervalSecFloat": 3.0, # Duration between updates for the Client
            "DumpLogListCountInt": 100, # Set the max row for the dump
            "DumpLogList": [], # Will be filled automatically
            "DumpLogListHashStr": None, # Will be filled automatically
            ##################
        },
        "ServerDict": {
            "AgentActivityLifetimeSecFloat": 1200.0, # Time in seconds to life for activity for the agent
            "AgentConnectionLifetimeSecFloat": 300.0, # Time in seconds to handle the open connection to the Agent
            "AgentLoopSleepSecFloat": 2.0, # Time in seconds to sleep between loops when check to send some activity to the agent
            "WorkingDirectoryPathStr": None, # Will be filled automatically
            "RequestTimeoutSecFloat": 300, # Time to handle request in seconds
            "ListenPort_": "Порт, по которому можно подключиться к демону",
            "ListenPort": 80,
            "ListenURLList": [
                {
                    "Description": "Local machine test",
                    "URL_": "Сетевое расположение сервера демона",
                    "URL": ""
                }
            ],
            "AccessUsers": { # Default - all URL is blocked
                "FlagCredentialsAsk": True, # Turn on Authentication
                "RuleDomainUserDict": {
                    # ("DOMAIN", "USER"): { !!!!! only in upper case !!!!
                    #   "MethodMatchURLBeforeList": [
                    #       {
                    #           "Method":"GET|POST",
                    #           "MatchType":"BeginWith|Contains|Equal|EqualCase",
                    #           "URL":"",
```

```
            #         "FlagAccessDefRequestGlobalAuthenticate": None, #Return bool
            #         "FlagAccess": True
            #       }
            #    ],
            #    "ControlPanelKeyAllowedList":[], # If empty - all is allowed
            #    "RoleHierarchyAllowedDict": {
            #      "Orchestrator":{
            #         "Controls": {
            #            "RestartOrchestrator": {}, # Feature to restart orchestrator on virtual machine
            #            "LookMachineScreenshots": {} # Feature to look machina screenshots
            #        },
            #         "RDPActive": { # Robot RDP active module
            #            "ListRead": {} # Access to read RDP session list
            #         }
            #      }
            #    }
            #  }
          },
          "RuleMethodMatchURLBeforeList": [  # General MethodMatchURL list (no domain/user)
            #     {
            #         "Method":"GET|POST",
            #         "MatchType":"BeginWith|Contains|Equal|EqualCase",
            #         "URL":"",
            #         "FlagAccessDefRequestGlobalAuthenticate": None, #Return bool
            #         "FlagAccess": True
            #      }
          ],
          "AuthTokensDict": {
            # "<AuthToken>":{"User":"", "Domain":"", "TokenDatetime":<Datetime>, "FlagDoNotExpire":True}
          }
        },
        "URLList": [  # List of available URLs with the orchestrator server
          # {
          #   "Method":"GET|POST",
          #   "URL": "/index", #URL of the request
          #   "MatchType": "", #"BeginWith|Contains|Equal|EqualCase",
          #   "ResponseFilePath": "", #Absolute or relative path
          #   "ResponseFolderPath": "", #Absolute or relative path
          #   "ResponseContentType": "", #HTTP Content-type
          #   "ResponseDefRequestGlobal": None #Function with str result
          # }
          {
            "Method": "GET",
            "URL": "/test/",  # URL of the request
            "MatchType": "BeginWith",  # "BeginWith|Contains|Equal|EqualCase",
            # "ResponseFilePath": "", #Absolute or relative path
            "ResponseFolderPath": "C:\Abs\Archive\scopeSrcUL\OpenRPA\Orchestrator\Settings",
            # Absolute or relative path
            # "ResponseContentType": "", #HTTP Content-type
            # "ResponseDefRequestGlobal": None #Function with str result
          }
        ],

      },
      "OrchestratorStart": {
        "DefSettingsUpdatePathList": [],
        # List of the .py files which should be loaded before init the algorythms
        "ActivityList": []
      },
      "SchedulerDict": {
        "CheckIntervalSecFloat": 5.0,  # Check interval in seconds
        "ActivityTimeList": [
          # {
          #   "TimeHH:MMStr": "22:23",  # Time [HH:MM] to trigger activity
          #   "WeekdayList": [0, 1, 2, 3, 4, 5, 6], #List of the weekday index when activity is applicable, Default [0,1,2,3,4,5,6]
          #   "ActivityList": [
          #               # {
          #             #   "Def":"DefAliasTest", # def link or def alias (look gSettings["Processor"]["AliasDefDict"])
          #             #   "ArgList":[1,2,3], # Args list
          #             #   "ArgDict":{"ttt":1,"222":2,"dsd":3} # Args dictionary
          #             #   "ArgGSettings": # Name of GSettings attribute: str (ArgDict) or index (for ArgList)
          #             #   "ArgLogger": None # Name of GSettings attribute: str (ArgDict) or index (for ArgList)
          #             #   "GUIDStr": ..., # GUID of the activity
          #             # },
          #   ],
          #   "GUID": None # Will be filled in Orchestrator automatically - is needed for detect activity completion
          # },
        ],
      },
      "ProcessorDict": {  # Has been changed. Newgeneral processor (one threaded) v.1.2.0
        "ActivityList": [  # List of the activities
          # {
```

```
        #    "Def":"DefAliasTest", # def link or def alias (look gSettings["Processor"]["AliasDefDict"])
        #    "ArgList":[1,2,3], # Args list
        #    "ArgDict":{"ttt":1,"222":2,"dsd":3} # Args dictionary
        #    "ArgGSettings": # Name of GSettings attribute: str (ArgDict) or index (for ArgList)
        #    "ArgLogger": None # Name of GSettings attribute: str (ArgDict) or index (for ArgList)
        #    "GUIDStr": ..., # GUID of the activity
        #},
      ],
      "AliasDefDict": {},  # Storage for def with Str alias. To use it see pyOpenRPA.Orchestrator.ControlPanel
      "CheckIntervalSecFloat": 1.0,  # Interval for check gSettings in ProcessorDict > ActivityList
      "ExecuteBool": True,  # Flag to execute thread processor
      "ThreadIdInt": None, # Technical field - will be setup when processor init
      "WarningExecutionMoreThanSecFloat": 60.0 # Push warning if execution more than n seconds
    },
    "ControlPanelDict": { # Old structure > CPDict
      "RefreshSeconds": 5,  # deprecated parameter
      "RobotList": [
        #{
        #    "RenderFunction": RenderRobotR01,
        #    "KeyStr": "TestControlPanelKey"
        #}
      ]
    },
    "CPDict": {
      # "CPKey": {"HTMLRenderDef":None, "JSONGeneratorDef":None, "JSInitGeneratorDef":None}
    },
    ##############
    "RobotRDPActive": {
      "RecoveryDict": {
        "CatchPeriodSecFloat": 1200, # Catch last 10 minutes
        "TriggerCountInt": 10, # Activate trigger if for the period orch will catch the reconnect RDP n times
        "DoDict": {
          "OSRemotePCRestart": True # Do powershell remote restart
        },
        "__StatisticsDict__": {
          # RDPSessionKeyStr : [time.time(), time.time()],
        }
      },
      "RDPList": {
        # "RDPSessionKey":{
        #    "Host": "77.77.22.22",  # Host address
        #    "Port": "3389",  # RDP Port
        #    "Login": "test",  # Login
        #    "Password": "test",  # Password
        #    "Screen": {
        #        "Width": 1680,  # Width of the remote desktop in pixels
        #        "Height": 1050,  # Height of the remote desktop in pixels
        #        # "640x480" or "1680x1050" or "FullScreen". If Resolution not exists set full screen
        #        "FlagUseAllMonitors": False,  # True or False
        #        "DepthBit": "32" # "32" or "24" or "16" or "15"
        #    },
        #    "SharedDriveList": ["c"],  # List of the Root sesion hard drives
        #    ####### Will updated in program ###############
        #    "SessionHex": "",  # Hex is created when robot runs
        #    "SessionIsWindowExistBool": False, # Flag if the RDP window is exist, old name "FlagSessionIsActive". Check every n seco
        #    "SessionIsWindowResponsibleBool": False, # Flag if RDP window is responsible (recieve commands). Check every nn sec
        #    "SessionIsIgnoredBool": False # Flag to ignore RDP window False - dont ignore, True - ignore
        #}
      },
      "ResponsibilityCheckIntervalSec": None,
      # Seconds interval when Robot check the RDP responsibility. if None - dont check
      "FullScreenRDPSessionKeyStr": None,
      # RDPSessionKeyStr of the current session which is full screened, None is no session in fullscreen
      "ActivityList": [
        # Technical Activity list for RobotRDPActive thread - equal to Main activity list, apply only RDP activity
        #{
        #    "DefNameStr":"test", # Function name in RobotRDPActive.Processor
        #    "ArgList":[1,2,3], # Args list
        #    "ArgDict":{"ttt":1,"222":2,"dsd":3} # Args dictionary
        #},
        #{
        #    "DefNameStr": "RDPSessionConnect",  # Function name in RobotRDPActive.Processor
        #    "ArgList": [],  # Args list
        #    "ArgDict": {"inRDPSessionKeyStr": "TestRDP", "inHostStr": "77.44.33.22", "inPortStr": "3389",
        #                "inLoginStr": "login", "inPasswordStr": "pass"}  # Args dictionary
        #},
        #{
        #    "DefNameStr": "RDPSessionDisconnect",  # Disconnect the RDP session without logoff. Function name in RobotRDPActive.
        #    "ArgList": [],  # Args list
        #    "ArgDict": {"inRDPSessionKeyStr": "TestRDP"}
        #},
        #{
```

```
            #"{
            #   "DefNameStr": "RDPSessionReconnect",  # Disconnect the RDP session without logoff. Function name in RobotRDPActive.
            #   "ArgList": [],  # Args list
            #   "ArgDict": {"inRDPSessionKeyStr": "TestRDP"}
            #}
         ]
      },
      ##############
      "FileManager": {
         "FileURLFilePathDict_help": "https://localhost:8081/filemanager/<file URL>. All FileURL s must be set in lowercase",
         "FileURLFilePathDict": {
            #"r01/report.xlsx": "C:\\RPA\\R01_IntegrationOrderOut\\Data\\Reestr_otgruzok.xlsx"
         }
      },
      "Logger": logging.getLogger("Orchestrator"),
      "StorageDict": {
         "Robot_R01_help": "Robot data storage in orchestrator env",
         "Robot_R01": {},
         "R01_OrchestratorToRobot": {"Test2": "Test2"}
      },
      "AgentDict": { # Will be filled when program runs
         #("HostNameUpperStr", "UserUpperStr"): { "IsListenBool": True, "QueueList": [] }
      },
      "AgentActivityReturnDict": { # Will be filled when programs run - fill result of the Activity execution on the agent
         # Key - Activity Item GUID str, Value {"Return": ..., "ReturnedByDatetime": datetime.datetime}
         # If key exists - def has been completed
      }
      # "HiddenIsOrchestratorInitialized" - will be inited when orchestrator will be initialized
   }

# Create full configuration for
def __AgentDictItemCreate__():
   return {"IsListenBool":False, "ConnectionCountInt":0, "ConnectionFirstQueueItemCountInt":0, "ActivityList":[]}

# Create full configuration for AgentActivityReturnDict
def __AgentActivityReturnDictItemCreate__(inReturn):
   return {"Return": inReturn, "ReturnedByDatetime": datetime.datetime.now()}

# Create full configuration for
def __UACClientAdminCreate__():
   lResultDict = {
      "pyOpenRPADict":{
         "CPKeyDict":{ # Empty dict - all access
            # "CPKeyStr"{
            #}
         },
         "RDPKeyDict":{ # Empty dict - all access
            #"RDPKeyStr"{
            #   "FullscreenBool": True,
            #   "IgnoreBool":True,
            #   "ReconnectBool": True
            #   "NothingBool": True # USe option if you dont want to give some access to the RDP controls
            #}
         },
         "AgentKeyDict": { # Empty dict - all access
            # "AgentKeyStr"{
            #}
         },
         "AdminDict":{ # Empty dict - all access
            "LogViewerBool":True, # Show log viewer on the web page
            "CMDInputBool":True, # Execute CMD on the server side and result to the logs
            "ScreenshotViewerBool":True, # Show button to look screenshots
            "RestartOrchestratorBool": True, # Restart orchestrator activity
            "RestartOrchestratorGITPullBool": True, # Turn off (RDP remember) orc + git pull + Turn on (rdp remember)
            "RestartPCBool": True, # Send CMD to restart pc
            "NothingBool":True # USe option if you dont want to give some access to the RDP controls
         },
         "ActivityDict": { # Empty dict - all access
            "ActivityListExecuteBool": True,  # Execute activity at the current thread
            "ActivityListAppendProcessorQueueBool": True  # Append activity to the processor queue
         }
      }

   }
   return lResultDict


# Init the log dump to WEB
# import pdb; pdb.set_trace()
##################################################
def LoggerDumpLogHandlerAdd(inLogger, inGSettingsClientDict):
   lL = inLogger
```

```
    if len(lL.handlers) == 0:
        mRobotLoggerFormatter = logging.Formatter('%(asctime)s - %(levelname)s - %(message)s')
    else:
        mRobotLoggerFormatter = lL.handlers[0].formatter
    mHandlerDumpLogList = LoggerHandlerDumpLogList.LoggerHandlerDumpLogList(inDict=inGSettingsClientDict,
        inKeyStr="DumpLogList", inHashKeyStr="DumpLogListHashStr", inRowCountInt=inGSettingsClientDict[
                                                "DumpLogListCountInt"])
    mHandlerDumpLogList.setFormatter(mRobotLoggerFormatter)
    lL.addHandler(mHandlerDumpLogList)

# inModeStr:
#   "BASIC" - create standart configuration
from pyOpenRPA.Orchestrator.Utils import LoggerHandlerDumpLogList
def Create(inModeStr="BASIC"):
    if inModeStr=="BASIC":
        lResult = __Create__() # Create settings
        # Создать файл логирования
        # add filemode="w" to overwrite
        if not os.path.exists("Reports"):
            os.makedirs("Reports")
        ##############################
        # Подготовка логгера Robot
        ##############################
        mRobotLogger = lResult["Logger"]
        mRobotLogger.setLevel(logging.INFO)
        # create the logging file handler
        mRobotLoggerFH = logging.FileHandler(
            "Reports\\" + datetime.datetime.now().strftime("%Y_%m_%d") + ".log")
        mRobotLoggerFormatter = logging.Formatter('%(asctime)s - %(levelname)s - %(message)s')
        mRobotLoggerFH.setFormatter(mRobotLoggerFormatter)
        # add handler to logger object
        mRobotLogger.addHandler(mRobotLoggerFH)
        ######################Add console output
        handler = logging.StreamHandler(sys.stdout)
        handler.setFormatter(mRobotLoggerFormatter)
        mRobotLogger.addHandler(handler)
        ################################################
        LoggerDumpLogHandlerAdd(inLogger=mRobotLogger, inGSettingsClientDict=lResult["Client"])
        #mHandlerDumpLogList = LoggerHandlerDumpLogList.LoggerHandlerDumpLogList(inDict=lResult["Client"],
        #                                                inKeyStr="DumpLogList",
        #                                                inHashKeyStr="DumpLogListHashStr",
        #                                                inRowCountInt=lResult["Client"][
        #                                                    "DumpLogListCountInt"])
        #mHandlerDumpLogList.setFormatter(mRobotLoggerFormatter)
        #mRobotLogger.addHandler(mHandlerDumpLogList)
    return lResult # return the result dict
```

Previous Next

4. How to use

You need to run orchestrator process?

```
if __name__ == "__main__": # New init way - allow run as module -m PyOpenRPA.Orchestrator
    from pyOpenRPA import Orchestrator  # Import orchestrator main
    gSettings = SettingsTemplate.Create(inModeStr="BASIC") # Create GSettings with basic configuration - no more config is available fr
    # Call the orchestrator main def
    Orchestrator.Orchestrator(inGSettings=gSettings)
```

If you need more configurations - so you can see here:

```
import psutil, datetime, logging, os, sys # stdout from logging

# Config settings
lPyOpenRPASourceFolderPathStr = r"..\Sources" # Path for test pyOpenRPA package

# Operations
if lPyOpenRPASourceFolderPathStr != "": sys.path.insert(0,os.path.abspath(os.path.join(lPyOpenRPASourceFolderPathStr))) # Path for t
# Start import after config the pyOpenRPA folder
from pyOpenRPA.Orchestrator import SettingsTemplate # Import functionallity

from pyOpenRPA import Orchestrator  # Import orchestrator main
#Run as administrator
if not Orchestrator.OrchestratorIsAdmin():
    Orchestrator.OrchestratorRerunAsAdmin()
    print(f"Orchestrator will be run as administrator!")
elif __name__ == "__main__": # New init way - allow run as module -m PyOpenRPA.Orchestrator
    gSettings = SettingsTemplate.Create(inModeStr="BASIC") # Create GSettings with basic configuration - no more config is available fro

    # TEST Add User ND - Add Login ND to superuser of the Orchestrator
    lUACClientDict = SettingsTemplate.__UACClientAdminCreate__()
    Orchestrator.UACUpdate(inGSettings=gSettings, inADLoginStr="ND", inADStr="", inADIsDefaultBool=True, inURLList=[], inRoleHierar
    # TEST Add User IMaslov - Add Login IMaslov to superuser of the Orchestrator
    Orchestrator.UACUpdate(inGSettings=gSettings, inADLoginStr="IMaslov", inADStr="", inADIsDefaultBool=True, inURLList=[])
    # TEST Add Supertoken for the all access between robots
    Orchestrator.UACSuperTokenUpdate(inGSettings=gSettings, inSuperTokenStr="1992-04-03-0643-ru-b4ff-openrpa52zzz")

    # Restore DUMP
    Orchestrator.OrchestratorSessionRestore(inGSettings=gSettings)

    # INFO Relative/Absolute import see below - after settings init
    # Template for import CP - Control Panels
    # ATTENTION - Pay attention to CP names! Orchestrator is one for the all control panels per one machine
    ## !!! For Absolute import control panels !!!
    #   try:
    #     sys.path.insert(0,os.path.abspath(os.path.join(r"..\ROBOT\Builds")))
    #     import pyRobot_CP
    #     pyRobot_CP.SettingsUpdate(inGSettings=gSettings)
    #   except Exception as e:
    #     gSettings["Logger"].exception(f"Exception when init CP. See below.")

    ## !!! For Relative import control panels !!!
    #   try:
    #     sys.path.insert(0,os.path.abspath(os.path.join(r"..\ROBOT\Builds")))
    #     from pyRobot_CP import ControlPanel
    #     ControlPanel.SettingsUpdate(inGSettings=gSettings)
    #   except Exception as e:
    #     gSettings["Logger"].exception(f"Exception when init CP. See below.")


    ## !!! For Relative import !!! CP Version Check
    try:
        sys.path.insert(0,os.path.abspath(os.path.join(r"")))
        from ControlPanel import CP_VersionCheck
        CP_VersionCheck.SettingsUpdate(inGSettings=gSettings)
    except Exception as e:
        gSettings["Logger"].exception(f"Exception when init CP. See below.")

    try:
        from ControlPanel import CP_Test
        CP_Test.SettingsUpdate(inGSettings=gSettings)
    except Exception as e:
        gSettings["Logger"].exception(f"Exception when init CP. See below.")

    # Call the orchestrator def
    Orchestrator.Orchestrator(inGSettings=gSettings, inDumpRestoreBool=False)
else:
    print("!WARNING! Current orchestrator settings do not support old type of the Orchestrator start. Use new Orchestrator type start (see
```

Previous Next

5. UAC - User Access Control

About

Orchestrator has mega feature - user access control (UAC). This feature allow you to manipulate access of the web UI for the all users!

If you need to give admin rights - you don't customize UAC dict.

If you need to give some little rights for user only for one robot in orchestrator web panel - you can set the following properties in UAC dict.

To work with UAC you can use defs `Orchestrator.UAC` group.

!!!ATTENTION!!!

Need practice example to work with UAC! - too hard to understand

UAC Dict for Orchestrator WEB UI rights

UAC Dict for pyOpenRPA Orchestrator WEB UI rights.

```
"pyOpenRPADict":{
    "CPKeyDict":{ # Empty dict - all access
        # "CPKeyStr"{
        # }
    },
    "RDPKeyDict":{ # Empty dict - all access
        #"RDPKeyStr"{
        #   "FullscreenBool": True,
        #   "IgnoreBool":True,
        #   "ReconnectBool": True
        #   "NothingBool": True # USe option if you dont want to give some access to the RDP controls
        # }
    },
    "AgentKeyDict": { # Empty dict - all access
        # "AgentKeyStr"{
        # }
    },
    "AdminDict":{ # Empty dict - all access
        "LogViewerBool":True, # Show log viewer on the web page
        "CMDInputBool":True, # Execute CMD on the server side and result to the logs
        "ScreenshotViewerBool":True, # Show button to look screenshots
        "RestartOrchestratorBool": True, # Restart orchestrator activity
        "RestartOrchestratorGITPullBool": True, # Turn off (RDP remember) orc + git pull + Turn on (rdp remember)
        "RestartPCBool": True, # Send CMD to restart pc
        "NothingBool":True # USe option if you dont want to give some access to the RDP controls
    },
    "ActivityDict": { # Empty dict - all access
        "ActivityListExecuteBool": True,  # Execute activity at the current thread
        "ActivityListAppendProcessorQueueBool": True  # Append activity to the processor queue
    }
}
```