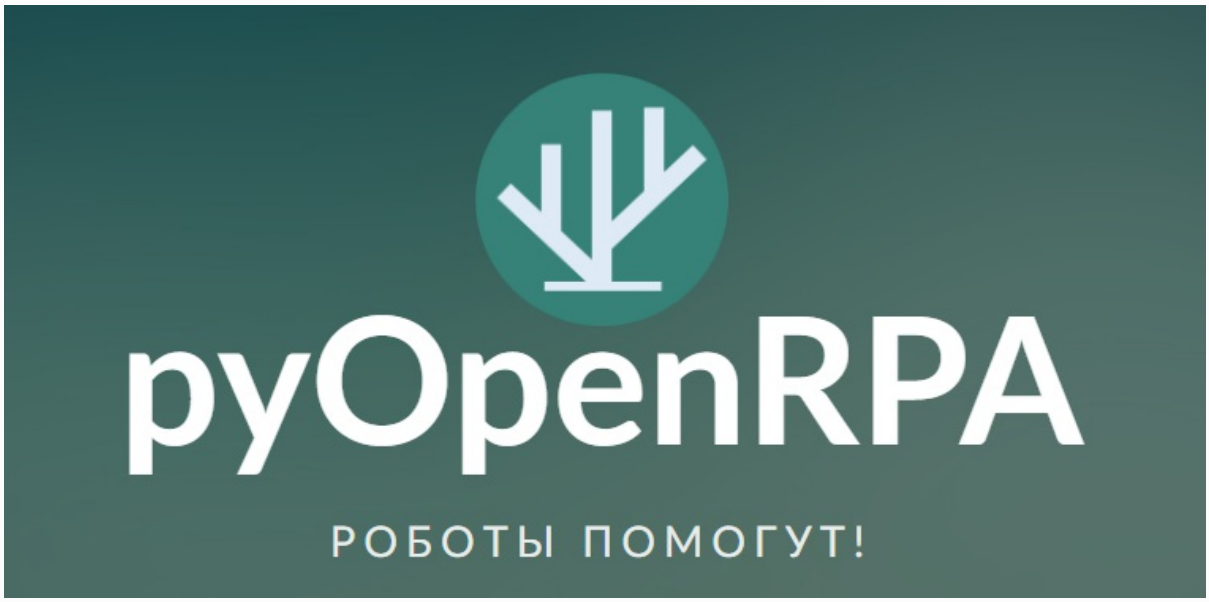
Портал документации pyOpenRPA



Дорогие коллеги!

Добро пожаловать на портал документации pyOpenRPA.

pyOpenRPA – это прогрессивная RPA платформа, которая позволяет сочетать в рамках одного проекта все перспективные технологии, такие как:

!ВНИМАНИЕ! Платформа pyOpenRPA включена в единый реестр российских программ для ЭВМ и БД

- OCR / NLP / NER: Распознавание изображений и текста
- CV: Компьютерное зрение
- AI: Искусственный интеллект
- ML: Машинное обучение
- BIGDATA: Большие данные
- VOICE & CHAT: Голосовые и чат-боты

Роботов pyOpenRPA невозможно отключить принудительно - управлять ими будете только вы!

Сегодня на рынке RPA явно выражены 4 проблемы:

- Программные роботы RPA являются настолько дорогими, что точно окупаются только в крупном бизнесе.
- Архитектура закрытых платформ RPA содержит очень ограниченный набор передовых технологий. А подключение таковых в платформу приходится выполнять только через выпуск новых версий.
- Возникающие дефекты закрытых платформ RPA блокируют выполнение всего робота.
- Зависимость RPA платформы от ключа активации вендора может заблокировать работу всех

ранее разработанных роботов (пример с санкциями от западных компаний, а помимо санкций возникают и другие причины).

Платформа ruOpenRPA глобально меняет рынок RPA и решает все вышеперечисленные проблемы. Более того, наше решение делает программную роботизацию RPA выгодной и эффективной с бизнес-эффекта всего от 10 тыс. рублей! Тем самым мы даём возможность технологического развития даже для микропредприятий! В компании ООО «ОПЕН РПА» мы создаем стандарты вендора / поставщика софта нового поколения. Предлагаем уникальный набор услуг для каждого клиента вплоть до реализации проекта под ключ с учетом максимальной экономической эффективности.

Наша открытость и ряд стратегических инициатив позволяют нам быть предельно прозрачными перед всеми участниками рынка. Предлагать индивидуальный набор услуг, нацеленный на решение проблем клиента.

С презентацией ruOpenRPA вы можете ознакомиться по ссылке: [СМОТРЕТЬ PDF] . Если возникнут вопросы, то вы всегда можете обратиться к нам (контакты см. в презентации).

ruOpenRPA - роботы помогут!

Иван Маслов, основатель и генеральный директор ООО «ОПЕН РПА» (ruOpenRPA LLC)

Лицензия ruOpenRPA

Лицензия ruOpenRPA разрешает бесплатное использование только для некоммерческих организаций и физических лиц (не ИП и не самозанятый). В остальных случаях требуется получение цифрового сертификата от правообладателя (ООО «ОПЕН РПА»).

Для коммерческого использования:

[СМОТРЕТЬ НА ГЛАВНОЙ СТРАНИЦЕ]

Используя ПО ruOpenRPA Вы осознаете свою ответственность в случаях нарушения лицензионного законодательства и совершения неправомерных действий.

Подробнее про лицензирование: 2. Лицензия & Контакты

О портале и программе

Дорогие коллеги!

На портале вы найдете все необходимое описание для решения любой задачи программной роботизации RPA.

Платформа ruOpenRPA является одной из самых производительных: скорость выполнения операций не уступает другим западным и Российским RPA аналогам. А в ряде случаев ruOpenRPA отрабатывает в 2 - 4 раза быстрее.

С 2019 года ruOpenRPA применяется в малом / среднем / крупном сегментах бизнеса для:

- отказа от западных RPA платформ;
- аудита финансовой и ИТ функций;
- прототипирования бизнес-процессов без изменения ИТ ландшафта;
- оптимизации нагрузки на коллег из классической автоматизации (legacy / back / front);

- реализации автотестов выпускаемых программ.

pyOpenRPA является программой с открытым исходным кодом. С исходными кодами можно ознакомиться по адресу: https://gitlab.com/UnicodeLabs/OpenRPA

Если у вас останутся вопросы, то вы всегда можете обратиться в центр поддержки клиентов pyOpenRPA. Контакты: 2. Лицензия & Контакты

pyOpenRPA - роботы помогут!

Портал состоит из следующих разделов:

- ОБЩЕЕ
- МОДУЛЬ РОБОТ
- МОДУЛЬ СТУДИЯ
- МОДУЛЬ ОРКЕСТРАТОР
- МОДУЛЬ АГЕНТ
- ИНСТРУМЕНТЫ

ОБЩЕЕ

В разделе описание общие положения, такие как: Описание структуры портала, инструкция по развертыванию и запуску pyOpenRPA, информация о правообладателе, контакты.

МОДУЛЬ РОБОТ

Модуль обеспечивает всю необходимую функциональность для создания любого программного робота RPA. Модуль робота поставляется в качестве библиотеки Python, что позволяет с легкостью интегрировать его в другие проекты перспективных технологий.

Содержит

- Уровень доступа к элементам локального приложения (win32, UI automation), и веб приложения
- Уровень доступа к текстовым каналам передачи данных (клавиатура, буфер обмена)
- Уровень доступа к графическим каналам передачи данных (мышь, экран)
- Уровень доступа к аудио каналам передачи данных (динамики, микрофон)

Подробное описание модуля: 1. Описание.

МОДУЛЬ СТУДИИ

Модуль является инструментом для отладки проектируемых узлов программного робота RPA. Запускается в качестве обособленного приложения в веб-браузере.

Основные возможности

- Поиск и запуск доступных действий над UI элементом
- Чтение и запись атрибутов UI элемента
- Визуальное прототипирование алгоритма
- Редактирование UI селекторов
- Поиск UI элементов по наведению мыши
- Поиск UI элементов в дереве UI объектов
- Автоматизированная генерация UI селекторов

Подробное описание модуля: 1. Описание.

МОДУЛЬ ОРКЕСТРАТОР

Модуль, который становится необходим, когда речь идет от нескольких работающих роботах. Запуск, остановка, контроль активности, проверка удаленных сессий, панель управления для бизнес-сотрудников - всё это можно реализовать в оркестраторе.

Основные возможности

- Запуск / пауза / безопасная остановка / принудительная остановка робота
- Интеллектуальное расписание
- Просмотр состояния графических сессий роботов через панель управления
- Удаленное администрирование сессий оркестратора и робота
- Среда отладки функциональности через панель управления оркестратора
- Консолидированное хранилище логов, доступное для просмотра через панель управления
- Ролевая модель разграничения доступа
- Функциональность очередей для координации роботов

МОДУЛЬ АГЕНТ

Модуль, который обеспечивает необходимую связь графической сессии робота с сессией оркестратора.

Основные возможности

- Выполнение команд на сессии робота и возвращение результата на источник запроса
- Получение скриншотов работы графической сессии
- Отправка / получение больших файлов (более 2 гб.)

Технические требования (минимальные)

ВЕРСИЯ v1.4.0

Ниже представлены минимальные технические требования для компонентов: Студия, Оркестратор, Робот, Агент

Windows: - ОС Windows 7 / 8 / 8.1 / 10 / 11 или Windows Server 2008 / 2012 / 2016 / 2019 / 2022 - RAM 2 Гб. - HDD 40 Гб. (без учета логов роботов / оркестратора) - CPU 1 x 1 ГГц

! ВНИМАНИЕ ! Требуется пакет KB2999226 если используется windows Vista/7/8/8.1/Server 2008/Server 2012 https://support.microsoft.com/ru-ru/help/2999226 ! ВНИМАНИЕ ! Использование компонента OpenCV возможно только ОС Windows 7/8/8/10 (Windows Server только с 2016) ! ВНИМАНИЕ ! В случае использования более 2-х графических сессий на 1й ОС Windows Server, необходимо приоретение лицензии Microsoft RDS.

Linux: - ОС Debian / Ubuntu / ALT Linux / Astra Linux / RED OS / ROSA Linux - RAM 1 Гб. - HDD 5 Гб. - CPU 1 x 1 ГГц

Технические требования (рекомендуемые)

ВЕРСИЯ v1.4.0

Ниже представлены рекомендуемые технические требования для компонентов: Студия,

Оркестратор, Робот, Агент

Windows: - ОС Windows 7 / 10 / 11 или Windows Server 2016 / 2019 / 2022 - RAM 16 Гб. - HDD 150 Гб. - CPU 6 x 2.2 ГГц.

! ВНИМАНИЕ ! Требуется пакет KB2999226 если используется windows Vista/7/8/8.1/Server 2008/Server 2012 https://support.microsoft.com/ru-ru/help/2999226 ! ВНИМАНИЕ ! Использование компонента OpenCV возможно только ОС Windows 7/8/8/10 (Windows Server только с 2016) ! ВНИМАНИЕ ! В случае использования более 2-х графических сессий на 1й ОС Windows Server, необходимо приоретение лицензии Microsoft RDS.

Linux: - ОС Debian / Ubuntu / ALT Linux / Astra Linux / RED OS / ROSA Linux - RAM 8 Гб. - HDD 150 Гб. - CPU 4 x 2.2 ГГц.

Структура репозитория

Описание каждой папки репозитория ruOpenRPA:

- **Agent**: Преднастроенный компонент ruOpenRPA Agent (Агент)
- **Orchestrator**: Преднастроенный компонент ruOpenRPA Orchestrator (Оркестратор)
- **Resources**: Сторонние ресурсы, используемые в ruOpenRPA.
- **Robot**: Преднастроенный компонент ruOpenRPA Robot (Робот)
- **Sources**: Исходные коды ruOpenRPA + исходные коды документации ruOpenRPA
- **Studio**: Преднастроенный компонент ruOpenRPA Studio (Студия)
- **Utils**: Вспомогательные инструменты для разработчика робота на ruOpenRPA
- **Wiki**: Документация, дополнительные материалы

Wiki структура

В папке Wiki представлены следующие материалы:

- RUS Портал документации в формате HTML [|ОТКРЫТЬ GITLAB|]
- RUS Портал документации в формате MarkDown [|ОТКРЫТЬ GITLAB|]
- RUS Портал документации в формате PDF [|ОТКРЫТЬ GITLAB|]
- RUS Практическое руководство по работе с Desktop UI [|ОТКРЫТЬ HABR|]; [|ОТКРЫТЬ GITLAB|]
- RUS Практическое руководство по работе с Web UI [|ОТКРЫТЬ HABR|]; [|ОТКРЫТЬ GITLAB|]
- ENG портал документации в формате HTML [|OPEN GITLAB|]
- ENG портал документации в формате MarkDown [|OPEN GITLAB|]
- ENG портал документации в формате PDF [|OPEN GITLAB|]

2. Лицензия & Контакты

ОГЛАВЛЕНИЕ

ОБЩЕЕ

МОДУЛЬ РОБОТ

МОДУЛЬ СТУДИЯ

МОДУЛЬ ОРКЕСТРАТОР

- AgentOSCMD()
- AgentOSFileBinaryDataBase64StrAppend()
- AgentOSFileBinaryDataBase64StrCreate()
- AgentOSFileBinaryDataBase64StrReceive()
- AgentOSFileBinaryDataBytesCreate()
- AgentOSFileBinaryDataReceive()
- AgentOSFileSend()
- AgentOSFileTextDataStrCreate()
- AgentOSFileTextDataStrReceive()
- AgentOSLogoff()
- AgentProcessWOExeUpperUserListGet()
- GSettingsGet()
- GSettingsKeyListValueAppend()
- GSettingsKeyListValueGet()
- GSettingsKeyListValueOperatorPlus()
- GSettingsKeyListValueSet()
- OSCMD()
- OSCredentialsVerify()
- OSLogoff()
- OSRemotePCRestart()
- OSRestart()
- Orchestrator()
- OrchestratorInitWait()
- OrchestratorIsAdmin()
- OrchestratorIsCredentialsAsk()
- OrchestratorIsInited()
- OrchestratorLoggerGet()
- OrchestratorPySearchInit()
- OrchestratorRerunAsAdmin()
- OrchestratorRestart()
- OrchestratorScheduleGet()
- OrchestratorSessionRestore()
- OrchestratorSessionSave()
- OrchestratorThreadStart()
- ProcessDefIntervalCall()
- ProcessIsStarted()
- ProcessListGet()
- ProcessStart()
- ProcessStop()
- ProcessorActivityItemAppend()
- ProcessorActivityItemCreate()
- ProcessorAliasDefCreate()
- ProcessorAliasDefUpdate()
- PythonStart()
- RDPSessionCMDRun()
- RDPSessionConnect()
- RDPSessionDisconnect()
- RDPSessionFileStoredRecieve()
- RDPSessionFileStoredSend()
- RDPSessionLogoff()

Быстрая навигация

.. v1.4.0 replace:: v1.4.0

Следующая

1. Первый запуск (Windows & Linux)

Готовы испытать всю мощь перспективных технологий?

Будет очень интересно - начинаем!

Первый запуск (Windows)

Для начала необходимо выполнить следующие действия:

- Скачать репозиторий pyOpenRPA с главной страницы https://pyopenrpa.ru/ (кнопка «Скачать»)
- Распаковать пакет куда угодно!

ВСЁ - Развертывание pyOpenRPA в Windows завершено! :)

Также доступен вариант по развертыванию pyOpenRPA в уже существующие проекты Python 3.7+ с помощью команды pip install pyOpenRPA.

Возникли трудности? Рекомендуем обратиться в центр поддержки клиентов pyOpenRPA. Контакты см. здесь: 2. Лицензия & Контакты

Первый запуск (Linux)

Для начала необходимо выполнить следующие действия:

Вариант 1 (быстрый) - Развернуть !терминальную! чистую ОС linux (например, Ubuntu) - Переместить папку git://Utils/Configure/ubuntu-kde на linux машину - Выполнить в терминале: ./configure.sh - Следовать инструкциям после запуска - Автоматическая настройка займет около 8 минут при наличии интернета со средней скоростью в 20 мб./сек.

Вариант 2 (длительный) - Скачать репозиторий pyOpenRPA с главной страницы https://pyopenrpa.ru/ (кнопка «Скачать») - Распаковать пакет куда угодно! - scrot: sudo apt-get scrot (компонент для извлечения скриншотов. pyOpenRPA.Robot.Screen) - xclip: sudo apt-get install xclip (компонент для работы с буфером обмена. pyOpenRPA.Robot.Clipboard) - setxkbmap: apt-get install x11-xkb-utils (компонент для взаимодействия с клавиатурой, https://command-not-found.com/setxkbmap)

ВСЁ - Развертывание pyOpenRPA в Linux завершено! :)

Также доступен вариант по развертыванию pyOpenRPA в уже существующие проекты Python 3.7+ с помощью команды pip install pyOpenRPA.

Возникли трудности? Рекомендуем обратиться в центр поддержки клиентов pyOpenRPA. Контакты см. здесь: 2. Лицензия & Контакты

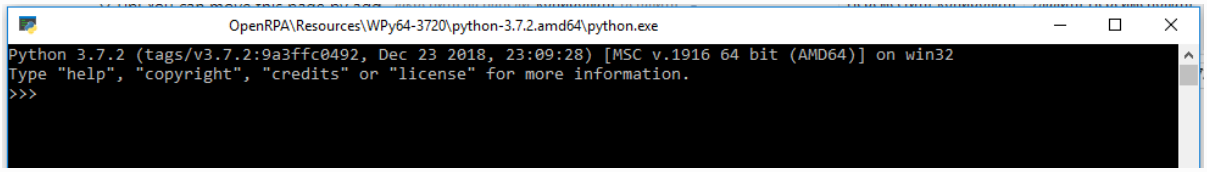
Проверить, что pyOpenRPA развернута корректно?

(Windows)

В папке pyOpenRPA запустить интерпретатор Python

- 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)

Платформа pyOpenRPA успешно развернута корректно. если интерпретаторы python 3.7.2 были запущены без проблем (см. скриншот).



Возникли трудности? Рекомендуем обратиться в центр поддержки клиентов pyOpenRPA. Контакты см. здесь: 2. Лицензия & Контакты

Проверить, что pyOpenRPA развернута корректно? (Linux)

В папке pyOpenRPA запустить интерпретатор Python

- x64 Python (./GIT/Resources/LPy64-3105/bin/python3.10)
- выполнить вход на страницу оркестратора (http://localhost:1024). Логин, установленный по-умолчанию: rpa00 (будет работать в том случае, если при настройке ./configure.sh указывали такую же УЗ). В остальных случаях требуется изменение настроек.

Платформа pyOpenRPA успешно развернута корректно, если интерпретатор python 3.10 был запущен без проблем.

Возникли трудности? Рекомендуем обратиться в центр поддержки клиентов pyOpenRPA. Контакты см. здесь: 2. Лицензия & Контакты

Быстрая навигация

- Сообщество pyOpenRPA (telegram)
- Сообщество pyOpenRPA (tenchat)
- Сообщество pyOpenRPA (вконтакте)
- Презентация pyOpenRPA
- Портал pyOpenRPA
- Репозиторий pyOpenRPA

.. v1.4.0 replace:: v1.4.0

Предыдущая Следующая

2. Лицензия & Контакты

Лицензия

Лицензия pyOpenRPA разрешает бесплатное использование только для некоммерческих организаций и физических лиц (не ИП и не самозанятый). В остальных случаях требуется получение цифрового сертификата от правообладателя (ООО «ОПЕН РПА»).

Получить, проверить сертификат, а также ознакомиться с текстом лицензионного соглашения Вы можете по адресу: https://pyopenrpa.ru/verification

pyOpenRPA не использует какие-либо инструменты физической блокировки функциональности своего ПО. По всем вопросам Вы можете обратиться к правообладателю, контакты см. по адресу: https://pyopenrpa.ru/Index/pyOpenRPA_product_service.pdf Используя ПО pyOpenRPA Вы осознаете свою ответственность в случаях нарушения лицензионного законодательства и совершения неправомерных действий.

ВНИМАНИЕ! НЕЗНАНИЕ ЗАКОНА НЕ ОСВОБОЖДАЕТ ОТ ОТВЕТСТВЕННОСТИ.

pyOpenRPA - роботы Вам помогут!

Автор

Маслов Иван Дмитриевич (с 2019г.)

Правообладатель

с 2022г. ООО «ОПЕН РПА», ОГРН 1227700251350, ИНН/КПП 9718191421/771801001 г. Москва: 125310, улица Муравская г. Санкт-Петербург: 197022, улица Льва Толстого

с 2019г. по 2021г. Маслов Иван Дмитриевич (с 2019г.)

Центр поддержки клиентов

У вас остались вопросы? Мы вам поможем!

- Телефон/WhatsApp: +7 (995) 233-45-31
- Почта: Support@pyOpenRPA.ru
- Телеграм: @pyOpenRPA_support :: Написать
- Портал: https://pyopenrpa.ru/
- Телеграм канал: @pyOpenRPA :: Присоединиться

Иван Маслов (генеральный директор ООО «ОПЕН РПА»)

- Телефон/WhatsApp: +7 (906) 722-39-25
- Почта: Ivan.Maslov@pyOpenRPA.ru

- Телеграм: @IvanMaslov :: Написать
- Портал: https://pyopenrpa.ru/
- Телеграм канал: @pyOpenRPA :: Присоединиться

Используемые сторонние компоненты (лицензионная чистота)

- WinPython, v3.7.1 32-bit & 64-bit, лицензия MIT https://github.com/winpython/winpython
- Selenium, v3.141.0, лицензия Apache 2.0 https://github.com/SeleniumHQ/selenium/blob/trunk/LICENSE
- pywinauto, v0.6.5, лицензия BSD 3-Clause https://github.com/pywinauto/pywinauto
- Semantic UI, v2.4.1, лицензия MIT https://github.com/Semantic-Org/Semantic-UI
- PyAutoGUI, v0.9.44, лицензия BSD 3-Clause https://github.com/asweigart/pyautogui
- keyboard, v0.13.3, лицензия MIT https://github.com/boppreh/keyboard)
- pywin32 (win32api), v228, Python Software Foundation лицензия (PSF) https://github.com/mhammond/pywin32
- WMI, v1.4.9, лицензия MIT, http://www.opensource.org/licenses/mit-license.php
- psutil, v5.6.2, лицензия BSD 3-Clause https://github.com/giampaolo/psutil/blob/master/LICENSE
- Pillow PIL, v6.0.0, лицензия HPND https://github.com/python-pillow/Pillow/blob/main/LICENSE
- requests, v2.22.0, лицензия Apache 2.0 https://github.com/psf/requests/blob/main/LICENSE
- JsRender, v1.0.2, лицензия MIT https://github.com/BorisMoore/jsrender/blob/master/MIT-LICENSE.txt
- Handlebars, v4.1.2, лицензия MIT, https://github.com/handlebars-lang/handlebars.js/blob/master/LICENSE
- jinja2, v2.11.2, лицензия BSD 3-Clause, https://github.com/pallets/jinja/blob/main/LICENSE.rst
- JupiterNotebook v6.1.4, лицензия BSD 3-Clause, https://github.com/jupyter/notebook/blob/main/LICENSE
- schedule, v1.1.0, лицензия MIT, https://github.com/dbader/schedule/blob/master/LICENSE.txt
- pyscreeze
- opencv
- numpy

Быстрая навигация

- Сообщество pyOpenRPA (telegram)
- Сообщество pyOpenRPA (tenchat)
- Сообщество pyOpenRPA (вконтакте)
- Презентация pyOpenRPA
- Портал pyOpenRPA
- Репозиторий pyOpenRPA

.. v1.4.0 replace:: v1.4.0

Предыдущая Следующая

1. Описание

Общее

Модуль обеспечивает всю необходимую функциональность для создания любого программного робота RPA. Модуль робота поставляется в качестве библиотеки Python, что позволяет с легкостью интегрировать его в другие проекты перспективных технологий.

Содержит

- **Уровень доступа к элементам локального приложения (win32, UI automation), и веб приложения**
  - UIDesktop: инструменты взаимодействия с элементами локального приложения (взаимодействие с ОС через протоколы win32, UI automation). Перейти к описанию функций: 2. Функции UIDesktop
  - UIWeb: инструменты взаимодействия с элементами веб приложения. Перейти к описанию функций: 3. Функции UIWeb
- **Уровень доступа к текстовым каналам передачи данных (клавиатура, буфер обмена)**
  - Keyboard: инструменты взаимодействия с клавиатурой. Перейти к описанию функций: 4. Функции Keyboard
  - Clipboard: инструменты взаимодействия с буфером обмена. Перейти к описанию функций: 5. Функции Clipboard
- **Уровень доступа к графическим каналам передачи данных (мышь, экран)**
  - Mouse: инструменты взаимодействия с мышью. Перейти к описанию функций: 6. Функции Mouse
  - Screen: инструменты взаимодействия с эраном рабочего стола. Перейти к описанию функций: 7. Функции Screen
- **Уровень доступа к звуковым каналам передачи данных (микрофон, динамик)**
  - Audio: инструменты взаимодействия с аудио. Перейти к описанию функций: 8. Функции Audio

Дорогие коллеги!

Мы знаем, что с pyOpenRPA вы сможете существенно улучшить качество вашего бизнеса. Платформа роботизации pyOpenRPA - это разработка, которая дает возможность делать виртуальных сотрудников (программных роботов RPA) выгодными, начиная от эффекта всего в **10 тыс. руб.** И управлять ими будете только Вы!

Если у вас останутся вопросы, то вы всегда можете обратиться в центр поддержки клиентов pyOpenRPA. Контакты: 2. Лицензия & Контакты

pyOpenRPA - роботы помогут!

Примеры

Ниже преставлен пример использования инструментов робота.

```
import time
```

```
import time
from pyOpenRPA.Robot import UIDesktop

# UIDesktop: Работа с 1С
lDemoBaseSelector = [{"title":"Запуск 1С:Предприятия","class_name":"V8TopLevelFrameTaxiStarter","backend"
lDemoBase = UIDesktop.UIOSelector_Get_UIO(lDemoBaseSelector)
lDemoBase.draw_outline()
time.sleep(2.0)
lRunBaseSelector = [{"title":"Запуск 1С:Предприятия","class_name":"V8TopLevelFrameTaxiStarter","backend":
lRunBase = UIDesktop.UIOSelector_Get_UIO(lRunBaseSelector)
lRunBase.draw_outline()
time.sleep(2.0)
lRunBase.click_input()

# ОТКРЫТЬ ЗАКАЗЫ ПОКУПАТЕЛЕЙ
lOrderNumberSelector = [{"title":"Управление нашей фирмой, редакция 1.6","class_name":"V8TopLevelFrameSDI
UIDesktop.UIOSelector_Get_UIO(lOrderNumberSelector).draw_outline()
UIDesktop.UIOSelector_Get_UIO(lOrderNumberSelector).double_click_input()

time.sleep(1.0)
lCommentSelector = [{"title":"Управление нашей фирмой, редакция 1.6","class_name":"V8TopLevelFrameSDI","b
UIDesktop.UIOSelector_Get_UIO(lCommentSelector).draw_outline()
UIDesktop.UIOSelector_Get_UIO(lCommentSelector).set_edit_text("Заказ исполнен роботом")

# UIWeb: Работа с браузером
# WIKI TO DO

# Keyboard: Взаимодействие с клавиатурой
import ctypes  # An included library with Python install.
from pyOpenRPA.Robot import Keyboard
from pyOpenRPA.Robot import Clipboard
Keyboard.send("win+r")
time.sleep(0.3)
Keyboard.write("cmd")
time.sleep(0.3)
Keyboard.send("enter")
time.sleep(0.6)
Keyboard.write("echo %time%")
time.sleep(0.3)
Keyboard.send("enter")
time.sleep(0.3)
Keyboard.send("ctrl+a")
time.sleep(0.6)
Clipboard.ClipboardSet("")
Keyboard.send("ctrl+c")
time.sleep(0.6)
lTextRaw = Clipboard.ClipboardGet()
lTimeStr = lTextRaw.split("\n")[-3]

def msg_box(title, text, style):
        return ctypes.windll.user32.MessageBoxW(0, text, title, style)
msg_box('Робот на клавиатуре', f'Робот извлек время из консоли: {lTimeStr}', 0)

# Mouse: Взаимодействие с мышью
from pyOpenRPA.Robot import Mouse
# Нарисовать букву Я
x = -50
y = 150
Mouse.mouseDown(x+100,y+0)
Mouse.moveTo(x+100,y+100)
Mouse.moveTo(x+100,y+50)
Mouse.moveTo(x+80,y+30)
Mouse.moveTo(x+100,y+0)
Mouse.moveTo(x+100,y+50)
Mouse.moveTo(x+80,y+100)
Mouse.mouseUp()

time.sleep(0.5)
# Нарисовать :)
x = 230
y = 150
Mouse.mouseDown(x+0,y+0)
Mouse.moveTo(x+0,y+75)
Mouse.mouseUp()

Mouse.mouseDown(x+75,y+0)
Mouse.moveTo(x+75,y+75)
Mouse.mouseUp()

Mouse.mouseDown(x-30,y+90)
Mouse.moveTo(x+40,y+130)
Mouse.moveTo(x+105,y+90)
Mouse.mouseUp()
```

```
Mouse.mouseUp()
```

Быстрая навигация

- Сообщество pyOpenRPA (telegram)
- Сообщество pyOpenRPA (tenchat)
- Сообщество pyOpenRPA (вконтакте)
- Презентация pyOpenRPA
- Портал pyOpenRPA
- Репозиторий pyOpenRPA

.. v1.4.0 replace:: v1.4.0

Предыдущая Следующая

2. Функции UIDesktop

Общее

Здесь представлено описание всех функций, которые используются для взаимодействия с UI интерфейсами локальных приложений.

Функции в модуле UIDesktop именуются по следующему принципу: <Входящий аргумент>_<действие>_<исходящий аргумент - если присутствует>

Термины и определения:

- **UIO:** Объект пользовательского интерфейса
- **UIOSelector:** Селектор (адрес) одного и/или более UIO объектов. Селектор представлен в формате списка (list) словарей (dict) атрибутивных критериев.

Описание функций

Описание каждой функции начинается с обозначения L-,W+, что означает, что функция не поддерживается в ОС Linux (L), но поддерживается в Windows (W)

Functions:

`BackendStr_GetTopLevelList_UIOInfo` ([inBackend])	L-,W+: Получить список UIOInfo словарей - процессы, котор
`CSS_To_UIOSelector` (inCSSSelector)	L+,W+:Выполнить конвертацию селектора из CSS в UIO
`GetFocused_UIO` ()	L-,W+: Получить UIO объект, на котором установлен фокус
`Get_OSBitnessInt` ()	L-,W+: Определить разрядность робота, в котором запускае
`PWASpecification_Get_PWAApplication` (...)	L-,W+: Получить значение атрибута backend по PWA (pywina
`PWASpecification_Get_UIO` (...)	L-,W+: Получить UIO объект по PWA (pywinauto) селектору.
`Selector_Convert_Selector` (inSelector, ...)	L+,W+: Техническая функция: Перевести селектор в заданны
`Selector_Get_Type` (inSelector)	L+,W+: Техническая функция: Определить тип селектора
`UIOEI_Convert_UIOInfo` (inElementInfo)	L-,W+: Техническая функция: Дообогащение словаря с парам
`UIOSelectorSecs_WaitAppear_Bool` (...[, ...])	L-,W+: Ожидать появление 1-го UIO объекта по заданному U
`UIOSelectorSecs_WaitDisappear_Bool` (...[, ...])	L-,W+: Ожидать исчезновение 1-го UIO объекта по заданном
`UIOSelectorUIOActivity_Get_ArgDict` (...)	L-,W+: Сформировать преднастроенный список/словарь арг
`UIOSelectorUIOActivity_Run_Dict` (...[, ...])	L-,W+: Выполнить активность inActionName над UIO объект
`UIOSelector_Exist_Bool` (inUIOSelector[, ...])	L-,W+: Проверить существование хотя бы 1-го UIO объекта
`UIOSelector_FocusHighlight` (inUIOSelector)	L-,W+: Установить фокус и подсветить на несколько секунд
`UIOSelector_GetChildList_UIOList` ([...])	L-,W+: Получить список дочерних UIO объектов по входяще
`UIOSelector_Get_BitnessInt` (inSpecificationList)	L-,W+: Определить разрядность приложения по UIO селект
`UIOSelector_Get_BitnessStr` (inSpecificationList)	L-,W+: Определить разрядность приложения по UIO селект
`UIOSelector_Get_UIO` (inSpecificationList[, ...])	L-,W+: Получить список UIO объект по UIO селектору.
`UIOSelector_Get_UIOActivityList` (inUIOSelector)	L-,W+: Получить список доступных действий/функций по UI
`UIOSelector_Get_UIOInfo` (inUIOSelector)	L-,W+: Получить свойства UIO объекта (element_info), по зад
`UIOSelector_Get_UIOInfoList` (inUIOSelector[, ...])	L-,W+: Техническая функция: Получить список параметров п
`UIOSelector_Get_UIOList` (inSpecificationList)	L-,W+: Получить список UIO объектов по UIO селектору
`UIOSelector_Highlight` (inUIOSelector)	L-,W+: Подсветить на несколько секунд на экране зеленой р
`UIOSelector_LevelInfo_List` (inUIOSelector[, ...])	L-,W+: Получить список свойств всех уровней до UI объекта
`UIOSelector_SafeOtherGet_Process` (inUIOSelector)	L-,W+: Получить процесс робота другой разрядности (если
`UIOSelector_SearchChildByMouse_UIO` (...)	L-,W+: Инициировать визуальный поиск UIO объекта с помо
`UIOSelector_SearchChildByMouse_UIOTree` (...)	L-,W+: Получить список уровней UIO объекта с указнием все
`UIOSelector_SearchProcessNormalize_UIOSelector` (...)	L-,W+: Нормализовать UIO селектор для дальнейшего испо.
`UIOSelector_SearchUIONormalize_UIOSelector` (...)	L-,W+: Нормализовать UIO селектор для дальнейшего испо.
`UIOSelector_To_CSS` (inUIOSelector)	L+,W+:Выполнить конвертацию селектора из UIO в CSS
`UIOSelector_To_XPATH` (inUIOSelector)	L+,W+:Выполнить конвертацию селектора из UIO в XPATH
`UIOSelector_TryRestore_Dict` (inSpecificationList)	L-,W+: Восстановить окно приложения на экране по UIO сел
`UIOSelectorsSecs_WaitAppear_List` (...[, ...])	L-,W+: Ожидать появление хотя бы 1-го / всех UIO объектов
`UIOSelectorsSecs_WaitDisappear_List` (...[, ...])	L-,W+: Ожидать исчезновение хотя бы 1-го / всех UIO объек
`UIOXY_SearchChild_ListDict` (inRootElement, ...)	L-,W+: Техническая функция: Получить иерархию вложенно
`UIO_FocusHighlight` (IWrapperObject[, colour, ...])	L-,W+: Установить фокус и выполнить подсветку UIO объек
`UIO_GetCtrlIndex_Int` (inElement)	L-,W+: Получить индекс UIO объекта inElement в списке род
`UIO_GetValue_Str` (inUIO)	Получить значение UI объекта по методу get_value.
`UIO_Highlight` (IWrapperObject[, colour, ...])	L-,W+: Выполнить подсветку UIO объекта на экране
`UIO_Search_UIOList` (inFilterStr, inFilterType)	L-,W+: Получить список UIO объектов по заданной строке.
`UIO_Search_UIOTree` (inFilterStr, inFilterType)	L-,W+: Получить список UIO объектов по заданной строке.
`XPATH_To_UIOSelector` (inXPATHSelector)	L+,W+:Выполнить конвертацию селектора из XPATH в UIO

pyOpenRPA.Robot.UIDesktop.BackendStr_GetTopLevelList_UIOInfo(*inBackend='win32'*) [исходный код]

L-,W+: Получить список UIOInfo словарей - процессы, которые запущены в рабочей сессии и готовы для взаимодействия с роботом через backend inBackend

```
# UIDesktop: Взаимодействие с UI объектами приложений
from pyOpenRPA.Robot import UIDesktop
lAppList = UIDesktop.BackendStr_GetTopLevelList_UIOInfo() # Очистить UIO селектор от недопустимых ключ
```

Параметры:

inBackend (*list*, *обязательный*) – вид backend, который планируется использовать для взаимодействия с UIO объектами

Результат:

список UIOInfo словарей

pyOpenRPA.Robot.UIDesktop.CSS_To_UIOSelector(*inCSSSelector*) [исходный код]

L+,W+:Выполнить конвертацию селектора из CSS в UIO

```
# UIDesktop: Взаимодействие с UI объектами приложений
from pyOpenRPA.Robot import UIDesktop
lCSSSelector = "[title='Запуск 1С:Предприятия'][class_name='V8TopLevelFrameTaxiStarter'][backend='uia'
lUIOSelector = UIDesktop.CSS_To_UIOSelector(inCSSSelector=lCSSSelector)
```

Параметры:

inCSSSelector (*str*, *обязательный*) – Селектор в формате CSS

pyOpenRPA.Robot.UIDesktop.GetFocused_UIO() [исходный код]

L-,W+: Получить UIO объект, на котором установлен фокус

```
# UIDesktop: Взаимодействие с UI объектами приложений
from pyOpenRPA.Robot import UIDesktop
lUIO = UIDesktop.GetFocused_UIO()
```

Результат:

UIO объект

pyOpenRPA.Robot.UIDesktop.Get_OSBitnessInt() [исходный код]

L-,W+: Определить разрядность робота, в котором запускается данная функция

```
from pyOpenRPA.Robot import UIDesktop
lRobotBitInt = UIDesktop.Get_OSBitnessInt() # Определить разрядность робота, в котором была вызвана эт
```

Результат:

64 (int) - разрядность приложения равна 64 битам; 32 (int) - разрядность приложения равна 32 битам

pyOpenRPA.Robot.UIDesktop.PWASpecification_Get_PWAApplication(*inControlSpecificationArray*) [исходный код]

L-,W+: Получить значение атрибута backend по PWA (pywinauto) селектору. Мы рекомендуем использовать метод UIOSelector_UIO_Get, так как UIO селектор обладает большей функциональностью.

```
# UIDesktop: Взаимодействие с UI объектами приложений
from pyOpenRPA.Robot import UIDesktop
# 1C: UIO Селектор выбора базы
lDemoBaseUIOSelector = [{"title":"Запуск 1С:Предприятия","class_name":"V8TopLevelFrameTaxiStarter","ba
lBackendStr = UIDesktop.PWASpecification_Get_PWAApplication(lDemoBaseUIOSelector) # Получить backend n
```

Параметры:

inControlSpecificationArray (*list*, *обязательный*) –

PWA селектор, который определяет критерии поиска UIO объекта Допустимые ключи PWA селектора:

- class_name содержимое атрибута class UIO объекта
- class_name_re содержимое атрибута class UIO объекта, которое удовлетворяет установленному рег. выражению
- process идентификатор процесса, в котором находится UIO объект
- title содержимое атрибута title UIO объекта
- title_re содержимое атрибута title UIO объекта, которое удовлетворяет установленному рег. выражению
- top_level_only признак поиска только на верхнем уровне приложения. По умолчанию True
- visible_only признак поиска только среди видимых UIO объектов. По умолчанию True
- enabled_only признак поиска только среди разблокированных UIO объектов. По

умолчанию False
- best_match содержимое атрибута title UIO объекта максимально приближено к заданному
- handle идентификатор handle искомого UIO объекта
- ctrl_index индекс UIO объекта среди всех дочерних объектов в списке родительского
- found_index индекс UIO объекта среди всех обнаруженных
- predicate_func пользовательская функция проверки соответсвия UIO элемента
- active_only признак поиска только среди активных UIO объектов. По умолчанию False
- control_id идентификатор control_id искомого UIO объекта
- control_type тип элемента (применимо, если backend == «uia»)
- auto_id идентификатор auto_id искомого UIO объекта (применимо, если backend == «uia»)
- framework_id идентификатор framework_id искомого UIO объекта (применимо, если backend == «uia»)
- backend вид технологии подключения к поиску UIO объекта («uia» или «win32»)

Результат:
«win32» или «uia»

pyOpenRPA.Robot.UIDesktop.PWASpecification_Get_UIO(*inControlSpecificationArray*) [исходный код]

L-,W+: Получить UIO объект по PWA (pywinauto) селектору. (https://pywinauto.readthedocs.io/en/latest/code/pywinauto.findwindows.html). Мы рекомендуем использовать метод UIOSelector_UIO_Get, так как UIO селектор обладает большей функциональностью.

```
# UIDesktop: Взаимодействие с UI объектами приложений
from pyOpenRPA.Robot import UIDesktop
# 1C: UIO Селектор выбора базы
lDemoBaseUIOSelector = [{"title":"Запуск 1С:Предприятия","class_name":"V8TopLevelFrameTaxiStarter","ba
lUIOObject = UIDesktop.PWASpecification_Get_UIO(lDemoBaseUIOSelector) # Получить UIO объект по PWA сел
```

Параметры:
inControlSpecificationArray (*list*, *обязательный*) –

PWA селектор, который определяет критерии поиска UIO объекта Допустимые ключи PWA селектора:

- class_name содержимое атрибута class UIO объекта
- class_name_re содержимое атрибута class UIO объекта, которое удовлетворяет установленному рег. выражению
- process идентификатор процесса, в котором находится UIO объект
- title содержимое атрибута title UIO объекта
- title_re содержимое атрибута title UIO объекта, которое удовлетворяет установленному рег. выражению
- top_level_only признак поиска только на верхнем уровне приложения. По умолчанию True
- visible_only признак поиска только среди видимых UIO объектов. По умолчанию True
- enabled_only признак поиска только среди разблокированных UIO объектов. По умолчанию False
- best_match содержимое атрибута title UIO объекта максимально приближено к заданному
- handle идентификатор handle искомого UIO объекта
- ctrl_index индекс UIO объекта среди всех дочерних объектов в списке родительского
- found_index индекс UIO объекта среди всех обнаруженных
- predicate_func пользовательская функция проверки соответсвия UIO элемента
- active_only признак поиска только среди активных UIO объектов. По умолчанию False
- control_id идентификатор control_id искомого UIO объекта
- control_type тип элемента (применимо, если backend == «uia»)
- auto_id идентификатор auto_id искомого UIO объекта (применимо, если backend == «uia»)
- framework_id идентификатор framework_id искомого UIO объекта (применимо, если backend == «uia»)
- backend вид технологии подключения к поиску UIO объекта («uia» или «win32»)

Результат:
UIO объект

pyOpenRPA.Robot.UIDesktop.Selector_Convert_Selector(*inSelector*, *inToTypeStr*) [исходный код]

L+,W+: Техническая функция: Перевести селектор в заданный тип. Доступно: UIO, CSS и XPATH

```
# UIDesktop: Взаимодействие с UI объектами приложений
from pyOpenRPA.Robot import UIDesktop
lToType = "XPATH"
lUIOSelector = [{"title":"Запуск 1С:Предприятия","class_name":"V8TopLevelFrameTaxiStarter","backend":"
lXPATHSelector = UIDesktop.Selector_Convert_Selector(inSelector=lUIOSelector, inToTypeStr=lToType)
```

Параметры:
- **inSelector** (*str или list*, *обязательный*) – Селектор, который необходимо преобразовать

- **inToTypeStr** (*str, обязательный*) – Тип, в который необходимо преобразовать селектор

Результат:
Селектор

pyOpenRPA.Robot.UIDesktop.Selector_Get_Type(*inSelector*) [исходный код]

L+,W+: Техническая функция: Определить тип селектора

```
# UIDesktop: Взаимодействие с UI объектами приложений
from pyOpenRPA.Robot import UIDesktop
lSelector = [{"title":"Запуск 1С:Предприятия","class_name":"V8TopLevelFrameTaxiStarter","backend":"uia
lType = UIDesktop.Selector_Get_Type(inSelector=lSelector)
```

Параметры:
inSelector (*str или list, обязательный*) – Селектор, тип которого необходимо определить

Результат:
Тип селектора (UIO, CSS или XPATH)

pyOpenRPA.Robot.UIDesktop.UIOEI_Convert_UIOInfo(*inElementInfo*) [исходный код]

L-,W+: Техническая функция: Дообогащение словаря с параметрами UIO объекта по заданному UIO.element_info

```
# UIDesktop: Взаимодействие с UI объектами приложений
from pyOpenRPA.Robot import UIDesktop
# 1C: UIO Селектор выбора базы
lDemoBaseUIOSelector = [{"title":"Запуск 1С:Предприятия","class_name":"V8TopLevelFrameTaxiStarter","ba
lUIO = UIDesktop.UIOSelector_Get_UIO(lDemoBaseUIOSelector) # Получить UIO объект по UIO селектору.
lUIOProcessInfoDict = UIDesktop.UIOEI_Convert_UIOInfo(lUIO.element_info)
```

Параметры:
inElementInfo (*object, обязательный*) – экземпляр класса UIO.element_info, для которого требуется дообогатить словарь с параметрами (в дальнейшем можно использовать как элемент UIO селектора).

Результат:
dict, пример:
{«title»:None,»rich_text»:None,»process_id»:None,»process»:None,»handle»:None,»class_name»:None,»control_type»:None,»control_i
{«left»:None,»top»:None,»right»:None,»bottom»:None}, „runtime_id“:None}

pyOpenRPA.Robot.UIDesktop.UIOSelectorSecs_WaitAppear_Bool(*inSpecificationList, inWaitSecs, inFlagRaiseException=True*) [исходный код]

L-,W+: Ожидать появление 1-го UIO объекта по заданному UIO селектору

!ВНИМАНИЕ! ДАННАЯ ФУНКЦИОНАЛЬНОСТЬ В АВТОМАТИЧЕСКОМ РЕЖИМЕ ПОДДЕРЖИВАЕТ ВСЕ РАЗРЯДНОСТИ ПРИЛОЖЕНИЙ (32|64), КОТОРЫЕ ЗАПУЩЕНЫ В СЕСИИ. PYTHON x64 ИМЕЕТ ВОЗМОЖНОСТЬ ВЗЗАИМОДЕЙСТВИЯ С x32 UIO ОБЪЕКТАМИ, НО МЫ РЕКОМЕНДУЕМ ДОПОЛНИТЕЛЬНО ИСПОЛЬЗОВАТЬ ИНТЕРПРЕТАТОР PYTHON x32 (ПОДРОБНЕЕ СМ. ФУНКЦИЮ Configure())

```
# UIDesktop: Взаимодействие с UI объектами приложений
from pyOpenRPA.Robot import UIDesktop
# 1C: UIO Селектор выбора базы
lDemoBaseUIOSelector = [{"title":"Запуск 1С:Предприятия","class_name":"V8TopLevelFrameTaxiStarter","ba
lDemoBaseUIOExistBool = UIDesktop.UIOSelectorSecs_WaitAppear_Bool(lDemoBaseUIOSelector) # Ожидать появ
```

Параметры:
- **inSpecificationList** (*list, обязательный*) – UIO селектор, который определяет критерии поиска UIO объекта
- **inWaitSecs** (*float, необязательный*) – Количество секунд, которые отвести на ожидание UIO объекта. По умолчанию 24 часа (86400 секунд)
- **inFlagRaiseException** (*bool, опциональный*) – True - формировать ошибку exception, если платформа не обнаружина ни одного UIO объекта по заданному UIO селектору. False - обратный случай (может привести к ошибочным результатам). По умолчанию True.

Результат:
True - UIO объект был обнаружен. False - обратная ситуациая

pyOpenRPA.Robot.UIDesktop.UIOSelectorSecs_WaitDisappear_Bool(*inSpecificationList, inWaitSecs, inFlagRaiseException=True*) [исходный код]

L-,W+: Ожидать исчезновение 1-го UIO объекта по заданному UIO селектору

!ВНИМАНИЕ! ДАННАЯ ФУНКЦИОНАЛЬНОСТЬ В АВТОМАТИЧЕСКОМ РЕЖИМЕ

ПОДДЕРЖИВАЕТ ВСЕ РАЗРЯДНОСТИ ПРИЛОЖЕНИЙ (32|64), КОТОРЫЕ ЗАПУЩЕНЫ В СЕСИИ. PYTHON x64 ИМЕЕТ ВОЗМОЖНОСТЬ ВЗЗАИМОДЕЙСТВИЯ С x32 UIO ОБЪЕКТАМИ, НО МЫ РЕКОМЕНДУЕМ ДОПОЛНИТЕЛЬНО ИСПОЛЬЗОВАТЬ ИНТЕРПРЕТАТОР PYTHON x32 (ПОДРОБНЕЕ СМ. ФУНКЦИЮ Configure())

```
# UIDesktop: Взаимодействие с UI объектами приложений
from pyOpenRPA.Robot import UIDesktop
# 1C: UIO Селектор выбора базы
lDemoBaseUIOSelector = [{"title":"Запуск 1С:Предприятия","class_name":"V8TopLevelFrameTaxiStarter","ba
lDemoBaseUIOExistBool = UIDesktop.UIOSelectorSecs_WaitDisappear_Bool(lDemoBaseUIOSelector) # Ожидать и
```

Параметры:

- **inSpecificationList** (*list*, *обязательный*) – UIO селектор, который определяет критерии поиска UIO объекта
- **inWaitSecs** (*float*, *необязательный*) – Количество секунд, которые отвести на исчезновение UIO объекта. По умолчанию 24 часа (86400 секунд)
- **inFlagRaiseException** (*bool*, *опциональный*) – True - формировать ошибку exception, если платформа не обнаружина ни одного UIO объекта по заданному UIO селектору. False - обратный случай (может привести к ошибочным результатам). По умолчанию True.

Результат:

True - UIO объект был обнаружен. False - обратная ситуациая

pyOpenRPA.Robot.UIDesktop.UIOSelectorUIOActivity_Get_ArgDict(*inUIOSelector*, *inActionStr*) [исходный код]

L-,W+: Сформировать преднастроенный список/словарь аргументов, передаваемый в функцию inActionStr, которая будет вызываться у объекта UIO, полученного по inUIOSelector

!ВНИМАНИЕ! ДАННАЯ ФУНКЦИОНАЛЬНОСТЬ В АВТОМАТИЧЕСКОМ РЕЖИМЕ ПОДДЕРЖИВАЕТ ВСЕ РАЗРЯДНОСТИ ПРИЛОЖЕНИЙ (32|64), КОТОРЫЕ ЗАПУЩЕНЫ В СЕСИИ. PYTHON x64 ИМЕЕТ ВОЗМОЖНОСТЬ ВЗЗАИМОДЕЙСТВИЯ С x32 UIO ОБЪЕКТАМИ, НО МЫ РЕКОМЕНДУЕМ ДОПОЛНИТЕЛЬНО ИСПОЛЬЗОВАТЬ ИНТЕРПРЕТАТОР PYTHON x32 (ПОДРОБНЕЕ СМ. ФУНКЦИЮ Configure())

```
# UIDesktop: Взаимодействие с UI объектами приложений
from pyOpenRPA.Robot import UIDesktop
# 1C: UIO Селектор выбора базы
lDemoBaseUIOSelector = [{"title":"Запуск 1С:Предприятия","class_name":"V8TopLevelFrameTaxiStarter","ba
lActivityResult = UIDesktop.UIOSelectorUIOActivity_Get_ArgDict(lDemoBaseUIOSelector, "type_keys")

#Возвращаемый результат
{'Selector': [{'title': 'UIDesktop.py - ORPA - Visual Studio Code',
'class_name': 'Chrome_WidgetWin_1',
'backend': 'uia'}],
'ArgList': ['keys', None, False, False, False, True, True, True],
'ArgDict': {'keys': None,
'pause': None,
'with_spaces': False,
'with_tabs': False,
'with_newlines': False,
'turn_off_numlock': True,
'set_foreground': True,
'vk_packet': True}}
```

Параметры:

- **inUIOSelector** (*list*, *обязательный*) – UIO селектор, который определяет UIO объект, для которого будет представлен перечень доступных активностей.
- **inActionStr** (*str*, *обязательный*) – Наименование метода, к которому выполнить подбор аргументов

pyOpenRPA.Robot.UIDesktop.UIOSelectorUIOActivity_Run_Dict(*inUIOSelector*, *inActionName*, *inFlagRaiseException=True*, *inArgumentList=None*, *inkwArgumentObject=None*) [исходный код]

L-,W+: Выполнить активность inActionName над UIO объектом, полученным с помощью UIO селектора inUIOSelector. Описание возможных активностей см. ниже.

!ВНИМАНИЕ! ДАННАЯ ФУНКЦИОНАЛЬНОСТЬ В АВТОМАТИЧЕСКОМ РЕЖИМЕ ПОДДЕРЖИВАЕТ ВСЕ РАЗРЯДНОСТИ ПРИЛОЖЕНИЙ (32|64), КОТОРЫЕ ЗАПУЩЕНЫ В СЕСИИ. PYTHON x64 ИМЕЕТ ВОЗМОЖНОСТЬ ВЗЗАИМОДЕЙСТВИЯ С x32 UIO ОБЪЕКТАМИ, НО МЫ РЕКОМЕНДУЕМ ДОПОЛНИТЕЛЬНО ИСПОЛЬЗОВАТЬ ИНТЕРПРЕТАТОР PYTHON x32 (ПОДРОБНЕЕ СМ. ФУНКЦИЮ Configure())

```
# UIDesktop: Взаимодействие с UI объектами приложений
from pyOpenRPA.Robot import UIDesktop
# 1C: UIO Селектор выбора базы
lDemoBaseUIOSelector = [{"title":"Запуск 1С:Предприятия","class_name":"V8TopLevelFrameTaxiStarter","ba
lActivityResult = UIDesktop.UIOSelectorUIOActivity_Run_Dict(lDemoBaseUIOSelector, "click") # выполнить
```

Параметры:

- **inUIOSelector** (*list*, *обязательный*) – UIO селектор, который определяет UIO объект, для

которого будет представлен перечень доступных активностей.

- **inActionName** (*str*, *обязательный*) – наименование активности, которую требуется выполнить над UIO объектом
- **inFlagRaiseException** (*bool*, *опциональный*) – True - формировать ошибку exception, если платформа не обнаружина ни одного UIO объекта по заданному UIO селектору. False - обратный случай (может привести к ошибочным результатам). По умолчанию True.
- **inArgumentList** (*list*, *необязательный*) – список передаваемых неименованных аргументов в функцию inActionName
- **inkwArgumentObject** (*dict*, *необязательный*) – словарь передаваемых именованных аргументов в функцию inActionName

Результат:

возвращает результат запускаемой функции с наименованием inActionName над UIO объектом

---

pyOpenRPA.Robot.UIDesktop.UIOSelector_Exist_Bool(*inUIOSelector*, *inFlagRaiseException=True*) [исходный код]

L-,W+: Проверить существование хотя бы 1-го UIO объекта по заданному UIO селектору

!ВНИМАНИЕ! ДАННАЯ ФУНКЦИОНАЛЬНОСТЬ В АВТОМАТИЧЕСКОМ РЕЖИМЕ ПОДДЕРЖИВАЕТ ВСЕ РАЗРЯДНОСТИ ПРИЛОЖЕНИЙ (32|64), КОТОРЫЕ ЗАПУЩЕНЫ В СЕСИИ. PYTHON x64 ИМЕЕТ ВОЗМОЖНОСТЬ ВЗАИМОДЕЙСТВИЯ С x32 UIO ОБЪЕКТАМИ, НО МЫ РЕКОМЕНДУЕМ ДОПОЛНИТЕЛЬНО ИСПОЛЬЗОВАТЬ ИНТЕРПРЕТАТОР PYTHON x32 (ПОДРОБНЕЕ СМ. ФУНКЦИЮ Configure())

```
# UIDesktop: Взаимодействие с UI объектами приложений
from pyOpenRPA.Robot import UIDesktop
# 1C: UIO Селектор выбора базы
lDemoBaseUIOSelector = [{"title":"Запуск 1С:Предприятия","class_name":"V8TopLevelFrameTaxiStarter","ba
lDemoBaseUIOExistBool = UIDesktop.UIOSelector_Exist_Bool(lDemoBaseUIOSelector) # Получить булевый резу
```

Параметры:

- **inUIOSelector** (*list*, *обязательный*) – UIO Селектор, который определяет критерии поиска UIO объектов
- **inFlagRaiseException** (*bool*, *опциональный*) – True - формировать ошибку exception, если платформа не обнаружина ни одного UIO объекта по заданному UIO селектору. False - обратный случай (может привести к ошибочным результатам). По умолчанию True.

Результат:

True - существует хотя бы 1 UIO объект. False - не существует ни одного UIO объекта по заданному UIO селектору

---

pyOpenRPA.Robot.UIDesktop.UIOSelector_FocusHighlight(*inUIOSelector*) [исходный код]

L-,W+: Установить фокус и подсветить на несколько секунд на экране зеленой рамкой UIO объект, который соответствует входящему UIO селектору inUIOSelector

!ВНИМАНИЕ! ДАННАЯ ФУНКЦИОНАЛЬНОСТЬ В АВТОМАТИЧЕСКОМ РЕЖИМЕ ПОДДЕРЖИВАЕТ ВСЕ РАЗРЯДНОСТИ ПРИЛОЖЕНИЙ (32|64), КОТОРЫЕ ЗАПУЩЕНЫ В СЕСИИ. PYTHON x64 ИМЕЕТ ВОЗМОЖНОСТЬ ВЗАИМОДЕЙСТВИЯ С x32 UIO ОБЪЕКТАМИ, НО МЫ РЕКОМЕНДУЕМ ДОПОЛНИТЕЛЬНО ИСПОЛЬЗОВАТЬ ИНТЕРПРЕТАТОР PYTHON x32 (ПОДРОБНЕЕ СМ. ФУНКЦИЮ Configure())

```
# UIDesktop: Взаимодействие с UI объектами приложений
from pyOpenRPA.Robot import UIDesktop
# 1C: UIO Селектор выбора базы
lDemoBaseUIOSelector = [{"title":"Запуск 1С:Предприятия","class_name":"V8TopLevelFrameTaxiStarter","ba
UIDesktop.UIOSelector_FocusHighlight(lDemoBaseUIOSelector) # Установить фокус и подсветить UIO объект
```

Параметры:

inUIOSelector (*list*, *обязательный*) – UIO селектор, который определяет UIO объект, для которого будет представлен перечень доступных активностей.

---

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

L-,W+: Получить список дочерних UIO объектов по входящему UIO селектору inUIOSelector.

!ВНИМАНИЕ! ДАННАЯ ФУНКЦИОНАЛЬНОСТЬ В АВТОМАТИЧЕСКОМ РЕЖИМЕ ПОДДЕРЖИВАЕТ ВСЕ РАЗРЯДНОСТИ ПРИЛОЖЕНИЙ (32|64), КОТОРЫЕ ЗАПУЩЕНЫ В СЕСИИ. PYTHON x64 ИМЕЕТ ВОЗМОЖНОСТЬ ВЗАИМОДЕЙСТВИЯ С x32 UIO ОБЪЕКТАМИ, НО МЫ РЕКОМЕНДУЕМ ДОПОЛНИТЕЛЬНО ИСПОЛЬЗОВАТЬ ИНТЕРПРЕТАТОР PYTHON x32 (ПОДРОБНЕЕ СМ. ФУНКЦИЮ Configure())

```
# UIDesktop: Взаимодействие с UI объектами приложений
from pyOpenRPA.Robot import UIDesktop
# 1C: UIO Селектор выбора базы
lDemoBaseUIOSelector = [{"title":"Запуск 1С:Предприятия","class_name":"V8TopLevelFrameTaxiStarter","ba
lUIOList = UIDesktop.UIOSelector_GetChildList_UIOList(lDemoBaseUIOSelector) # Получить список дочерних
```

Параметры:

- **inUIOSelector** (*list*, *обязательный*) – родительский UIO объект, полученный ранее с помощью UIO селектора.
- **inBackend** (*str*, *необязательный*) – вид backend «win32» или «uia». По умолчанию mDefaultPywinautoBackend («win32»)

Результат:

список дочерних UIO объектов

pyOpenRPA.Robot.UIDesktop.UIOSelector_Get_BitnessInt(*inSpecificationList*) [исходный код]

L-,W+: Определить разрядность приложения по UIO селектору. Вернуть результат в формате целого числа (64 или 32)

```
# UIDesktop: Взаимодействие с UI объектами приложений
from pyOpenRPA.Robot import UIDesktop
# 1C: UIO Селектор выбора базы
lDemoBaseUIOSelector = [{"title":"Запуск 1С:Предприятия","class_name":"V8TopLevelFrameTaxiStarter","ba
lDemoBaseBitInt = UIDesktop.UIOSelector_Get_BitnessInt(lDemoBaseUIOSelector) # Определить разрядность
```

Параметры:

inSpecificationList (*list*, *обязательный*) – UIO селектор, который определяет критерии поиска UIO объекта

Результат:

None - UIO объект не обнаружен; 64 (int) - разрядность приложения равна 64 битам; 32 (int) - разрядность приложения равна 32 битам

pyOpenRPA.Robot.UIDesktop.UIOSelector_Get_BitnessStr(*inSpecificationList*) [исходный код]

L-,W+: Определить разрядность приложения по UIO селектору. Вернуть результат в формате строки («64» или «32»)

```
# UIDesktop: Взаимодействие с UI объектами приложений
from pyOpenRPA.Robot import UIDesktop
# 1C: UIO Селектор выбора базы
lDemoBaseUIOSelector = [{"title":"Запуск 1С:Предприятия","class_name":"V8TopLevelFrameTaxiStarter","ba
lDemoBaseBitStr = UIDesktop.UIOSelector_Get_BitnessStr(lDemoBaseUIOSelector) # Определить разрядность
```

Параметры:

inSpecificationList (*list*, *обязательный*) – UIO селектор, который определяет критерии поиска UIO объекта

Результат:

None - UIO объект не обнаружен; «64» (str) - разрядность приложения равна 64 битам; «32» (str) - разрядность приложения равна 32 битам

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

L-,W+: Получить список UIO объект по UIO селектору. Если критериям UIO селектора удовлетворяет несколько UIO объектов - вернуть первый из списка

```
# UIDesktop: Взаимодействие с UI объектами приложений
from pyOpenRPA.Robot import UIDesktop
# 1C: UIO Селектор выбора базы
lDemoBaseUIOSelector = [{"title":"Запуск 1С:Предприятия","class_name":"V8TopLevelFrameTaxiStarter","ba
lDemoBaseUIOList = UIDesktop.UIOSelector_Get_UIO(lDemoBaseUIOSelector) #Получить 1-й UIO объект, котор
```

Параметры:

- **inSpecificationList** (*list*, *обязательный*) – UIO Селектор, который определяет критерии поиска UI элементов
- **inElement** (*UIO объект*, *опциональный*) – Родительский элемент, от которого выполнить поиск UIO объектов по заданному UIO селектору. Если аргумент не задан, платформа выполнит поиск UIO объектов среди всех доступных приложений windows, которые запущены на текущей сессии
- **inFlagRaiseException** (*bool*, *опциональный*) – True - формировать ошибку exception, если платформа не обнаружина ни одного UIO объекта по заданному UIO селектору. False - обратный случай. По умолчанию True

Результат:

UIO объект, которые удовлетворяют условиям UIO селектора, или None

pyOpenRPA.Robot.UIDesktop.UIOSelector_Get_UIOActivityList(*inUIOSelector*, *inFlagRaiseException=True*)
[исходный код]

L-,W+: Получить список доступных действий/функций по UIO селектору inUIOSelector. Описание возможных активностей см. ниже.

!ВНИМАНИЕ! ДАННАЯ ФУНКЦИОНАЛЬНОСТЬ В АВТОМАТИЧЕСКОМ РЕЖИМЕ ПОДДЕРЖИВАЕТ ВСЕ РАЗРЯДНОСТИ ПРИЛОЖЕНИЙ (32|64), КОТОРЫЕ ЗАПУЩЕНЫ В СЕСИИ. PYTHON x64 ИМЕЕТ ВОЗМОЖНОСТЬ ВЗЗАИМОДЕЙСТВИЯ С x32 UIO ОБЪЕКТАМИ, НО МЫ РЕКОМЕНДУЕМ ДОПОЛНИТЕЛЬНО ИСПОЛЬЗОВАТЬ ИНТЕРПРЕТАТОР PYTHON x32 (ПОДРОБНЕЕ СМ. ФУНКЦИЮ Configure())

```
# UIDesktop: Взаимодействие с UI объектами приложений
from pyOpenRPA.Robot import UIDesktop
# 1C: UIO Селектор выбора базы
lDemoBaseUIOSelector = [{"title":"Запуск 1С:Предприятия","class_name":"V8TopLevelFrameTaxiStarter","ba
lActivityList = UIDesktop.UIOSelector_Get_UIOActivityList(lDemoBaseUIOSelector) # Получить список акти
```

Параметры:

- **inUIOSelector** (*list*, *обязательный*) – UIO селектор, который определяет UIO объект, для которого будет представлен перечень доступных активностей.
- **inFlagRaiseException** (*bool*, *опциональный*) – True - формировать ошибку exception, если платформа не обнаружина ни одного UIO объекта по заданному UIO селектору. False - обратный случай (может привести к ошибочным результатам). По умолчанию True.

pyOpenRPA.Robot.UIDesktop.UIOSelector_Get_UIOInfo(*inUIOSelector*) [исходный код]

L-,W+: Получить свойства UIO объекта (element_info), по заданному UIO селектору. Ниже представлен перечень возвращаемых свойств.

Для backend = win32:

- automation_id (int)
- class_name (str)
- control_id (int)
- control_type (str)
- full_control_type (str)
- enabled (bool)
- handle (int)
- name (str)
- parent (object/UIO)
- process_id (int)
- rectangle (object/rect)
- rich_text (str)
- visible (bool)

Для backend = uia:

- automation_id (int)
- class_name (str)
- control_id (int)
- control_type (str)
- enabled (bool)
- framework_id (int)
- handle (int)
- name (str)
- parent (object/UIO)
- process_id (int)
- rectangle (object/rect)
- rich_text (str)
- runtime_id (int)
- visible (bool)

!ВНИМАНИЕ! ДАННАЯ ФУНКЦИОНАЛЬНОСТЬ В АВТОМАТИЧЕСКОМ РЕЖИМЕ ПОДДЕРЖИВАЕТ ВСЕ РАЗРЯДНОСТИ ПРИЛОЖЕНИЙ (32|64), КОТОРЫЕ ЗАПУЩЕНЫ В СЕСИИ. PYTHON x64 ИМЕЕТ ВОЗМОЖНОСТЬ ВЗЗАИМОДЕЙСТВИЯ С x32 UIO ОБЪЕКТАМИ, НО МЫ РЕКОМЕНДУЕМ ДОПОЛНИТЕЛЬНО ИСПОЛЬЗОВАТЬ ИНТЕРПРЕТАТОР PYTHON x32 (ПОДРОБНЕЕ СМ. ФУНКЦИЮ Configure())

```
# UIDesktop: Взаимодействие с UI объектами приложений
from pyOpenRPA.Robot import UIDesktop
# 1C: UIO Селектор выбора базы
lDemoBaseUIOSelector = [{"title":"Запуск 1С:Предприятия","class_name":"V8TopLevelFrameTaxiStarter","ba
lUIOElementInfoDict = UIDesktop.UIOSelector_Get_UIOInfo(lDemoBaseUIOSelector) #Получить свойства над U
```

Параметры:

inUIOSelector (*list*, *обязательный*) – UIO селектор, который определяет UIO объект, для которого будет представлен перечень доступных активностей.

Результат:

словарь свойств element_info: Пример {«control_id»: ..., «process_id»: ...}

pyOpenRPA.Robot.UIDesktop.UIOSelector_Get_UIOInfoList(*inUIOSelector*, *inElement=None*, *inFlagRaiseException=True*) [исходный код]

L-,W+: Техническая функция: Получить список параметров последних уровней UIO селектора по UIO объектам, которые удовлетворяют входящим inUIOSelector, поиск по которым будет производится от уровня inElement.

!ВНИМАНИЕ! ДАННАЯ ФУНКЦИОНАЛЬНОСТЬ В АВТОМАТИЧЕСКОМ РЕЖИМЕ ПОДДЕРЖИВАЕТ ВСЕ РАЗРЯДНОСТИ ПРИЛОЖЕНИЙ (32|64), КОТОРЫЕ ЗАПУЩЕНЫ В СЕСИИ. PYTHON x64 ИМЕЕТ ВОЗМОЖНОСТЬ ВЗЗАИМОДЕЙСТВИЯ С x32 UIO ОБЪЕКТАМИ, НО МЫ РЕКОМЕНДУЕМ ДОПОЛНИТЕЛЬНО ИСПОЛЬЗОВАТЬ ИНТЕРПРЕТАТОР PYTHON x32 (ПОДРОБНЕЕ СМ. ФУНКЦИЮ Configure())

```
# UIDesktop: Взаимодействие с UI объектами приложений
from pyOpenRPA.Robot import UIDesktop
# 1C: UIO Селектор выбора базы
lDemoBaseUIOSelector = [{"title":"Запуск 1С:Предприятия","class_name":"V8TopLevelFrameTaxiStarter","ba
lUIOInfoList = UIDesktop.UIOSelector_Get_UIOInfoList(lDemoBaseUIOSelector) # Получить словарь параметр
```

Параметры:

- **inUIOSelector** (*list*, *обязательный*) – UIO селектор, который определяет UIO объект, для которого будет произведено извлечение всех атрибутов на всех уровнях.
- **inElement** (*UIO объект*, *необязательный*) – UIO объект, от которого выполнить поиск дочерних UIO объектов по UIO селектору inUIOSelector. По умолчанию None - поиск среди всех приложений.
- **inFlagRaiseException** (*bool*, *опциональный*) – True - формировать ошибку exception, если платформа не обнаружина ни одного UIO объекта по заданному UIO селектору. False - обратный случай (может привести к ошибочным результатам). По умолчанию True.

Результат:

dict, пример:
{«title»:None,»rich_text»:None,»process_id»:None,»process»:None,»handle»:None,»class_name»:None,»control_type»:None,»control_i
{«left»:None,»top»:None,»right»:None,»bottom»:None}, „runtime_id“:None}

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

L-,W+: Получить список UIO объектов по UIO селектору

```
# UIDesktop: Взаимодействие с UI объектами приложений
from pyOpenRPA.Robot import UIDesktop
# 1C: UIO Селектор выбора базы
lDemoBaseUIOSelector = [{"title":"Запуск 1С:Предприятия","class_name":"V8TopLevelFrameTaxiStarter","ba
lDemoBaseUIOList = UIDesktop.UIOSelector_Get_UIOList(lDemoBaseUIOSelector) #Получить список UIO объект
```

Параметры:

- **inSpecificationList** (*list*, *обязательный*) – UIO Селектор, который определяет критерии поиска UI элементов
- **inElement** (*UIO объект*, *опциональный*) – Родительский элемент, от которого выполнить поиск UIO объектов по заданному UIO селектору. Если аргумент не задан, платформа выполнит поиск UIO объектов среди всех доступных приложений windows, которые запущены на текущей сессии
- **inFlagRaiseException** (*bool*, *опциональный*) – True - формировать ошибку exception, если платформа не обнаружина ни одного UIO объекта по заданному UIO селектору. False - обратный случай (может привести к ошибочным результатам). По умолчанию True.

Результат:

Список UIO объектов, которые удовлетворяют условиям UIO селектора

pyOpenRPA.Robot.UIDesktop.UIOSelector_Highlight(*inUIOSelector*) [исходный код]

L-,W+: Подсветить на несколько секунд на экране зеленой рамкой UIO объект, который

соответствует входящему UIO селектору inUIOSelector

!ВНИМАНИЕ! ДАННАЯ ФУНКЦИОНАЛЬНОСТЬ В АВТОМАТИЧЕСКОМ РЕЖИМЕ ПОДДЕРЖИВАЕТ ВСЕ РАЗРЯДНОСТИ ПРИЛОЖЕНИЙ (32|64), КОТОРЫЕ ЗАПУЩЕНЫ В СЕСИИ. PYTHON x64 ИМЕЕТ ВОЗМОЖНОСТЬ ВЗЗАИМОДЕЙСТВИЯ С x32 UIO ОБЪЕКТАМИ, НО МЫ РЕКОМЕНДУЕМ ДОПОЛНИТЕЛЬНО ИСПОЛЬЗОВАТЬ ИНТЕРПРЕТАТОР PYTHON x32 (ПОДРОБНЕЕ СМ. ФУНКЦИЮ Configure())

```
# UIDesktop: Взаимодействие с UI объектами приложений
from pyOpenRPA.Robot import UIDesktop
# 1C: UIO Селектор выбора базы
lDemoBaseUIOSelector = [{"title":"Запуск 1С:Предприятия","class_name":"V8TopLevelFrameTaxiStarter","ba
UIDesktop.UIOSelector_Highlight(lDemoBaseUIOSelector) # Подсветить UIO объект по UIO селектору
```

Параметры:

inUIOSelector (*list*, *обязательный*) – UIO селектор, который определяет UIO объект, для которого будет представлен перечень доступных активностей.

pyOpenRPA.Robot.UIDesktop.UIOSelector_LevelInfo_List(*inUIOSelector*, *inBackend='win32'*) [исходный код]

L-,W+: Получить список свойств всех уровней до UI объекта, обнаруженного с помощью селектора inUIOSelector.

!ВНИМАНИЕ! ДАННАЯ ФУНКЦИОНАЛЬНОСТЬ В АВТОМАТИЧЕСКОМ РЕЖИМЕ ПОДДЕРЖИВАЕТ ВСЕ РАЗРЯДНОСТИ ПРИЛОЖЕНИЙ (32|64), КОТОРЫЕ ЗАПУЩЕНЫ В СЕСИИ. PYTHON x64 ИМЕЕТ ВОЗМОЖНОСТЬ ВЗЗАИМОДЕЙСТВИЯ С x32 UIO ОБЪЕКТАМИ, НО МЫ РЕКОМЕНДУЕМ ДОПОЛНИТЕЛЬНО ИСПОЛЬЗОВАТЬ ИНТЕРПРЕТАТОР PYTHON x32 (ПОДРОБНЕЕ СМ. ФУНКЦИЮ Configure())

```
# UIDesktop: Взаимодействие с UI объектами приложений
from pyOpenRPA.Robot import UIDesktop
# 1C: UIO Селектор выбора базы
lDemoBaseUIOSelector = [{"title":"Запуск 1С:Предприятия","class_name":"V8TopLevelFrameTaxiStarter","ba
lUIOList = UIDesktop.UIOSelector_LevelInfo_List(lDemoBaseUIOSelector) # Получить список свойств каждог
```

Параметры:

- **inUIOSelector** (*list*, *обязательный*) – Селектор на UIO объект (CSS | XPATH | UIO).
- **inBackend** (*str*, *необязательный*) – вид backend «win32» или «uia». По умолчанию mDefaultPywinautoBackend («win32»)

Результат:

список дочерних UIO объектов

pyOpenRPA.Robot.UIDesktop.UIOSelector_SafeOtherGet_Process(*inUIOSelector*) [исходный код]

L-,W+: Получить процесс робота другой разрядности (если приложение UIO объекта выполняется в другой разрядности). Функция возвращает None, если разрядность робота совпадает с разрядностью приложения UIO объекта, либо если при инициализации робота не устанавливался интерпретатор другой разрядности.

```
# UIDesktop: Взаимодействие с UI объектами приложений
from pyOpenRPA.Robot import UIDesktop
# 1C: UIO Селектор выбора базы
lDemoBaseUIOSelector = [{"title":"Запуск 1С:Предприятия","class_name":"V8TopLevelFrameTaxiStarter","ba
lOtherBitnessProcess = UIDesktop.UIOSelector_SafeOtherGet_Process(lDemoBaseUIOSelector) # Вернуть проц
```

Параметры:

inUIOSelector (*list*, *обязательный*) – UIO селектор, который определяет критерии поиска UIO объекта

Результат:

Процесс робота схожей разрядности

pyOpenRPA.Robot.UIDesktop.UIOSelector_SearchChildByMouse_UIO(*inElementSpecification*) [исходный код]

L-,W+: Инициировать визуальный поиск UIO объекта с помощью указателя мыши. При наведении указателя мыши UIO объект выделяется зеленой рамкой. Остановить режим поиска можно с помощью зажима клавиши ctrl left на протяжении нескольких секунд. После этого в веб окне студии будет отображено дерево расположения искомого UIO объекта.

```
# UIDesktop: Взаимодействие с UI объектами приложений
from pyOpenRPA.Robot import UIDesktop
# 1C: UIO Селектор выбора базы
lDemoBaseUIOSelector = [{"title":"Запуск 1С:Предприятия","class_name":"V8TopLevelFrameTaxiStarter","ba
lUIO = UIDesktop.UIOSelector_SearchChildByMouse_UIO(lDemoBaseUIOSelector) # Инициировать поиск дочерне
```

Параметры:

inElementSpecification (*list*, *обязательный*) – UIO селектор, который определяет критерии поиска родительского UIO объекта, в котором будет производиться поиск дочернего UIO объекта

Результат:

UIO объект или None (если UIO не был обнаружен)

pyOpenRPA.Robot.UIDesktop.UIOSelector_SearchChildByMouse_UIOTree(*inUIOSelector*, *inWaitBeforeSec=0.0*)

[исходный код]

L-,W+: Получить список уровней UIO объекта с указнием всех имеющихся атрибутов по входящему UIO селектору.

ОБНОВЛЕНИЕ 1.4.0: Функция обновлена под использование в новой версии студии

ОБНОВЛЕНИЕ 1.4.0: ПОСЛЕ ОТРАБОТКИ ВОЗВРАЩАЕТ ФОКУС НА ТО ОКНО, КОТОРОЕ БЫЛО ПРИ ИНИЦИАЛИЗАЦИИ ФУНКЦИИ

!ВНИМАНИЕ! ДАННАЯ ФУНКЦИОНАЛЬНОСТЬ В АВТОМАТИЧЕСКОМ РЕЖИМЕ ПОДДЕРЖИВАЕТ ВСЕ РАЗРЯДНОСТИ ПРИЛОЖЕНИЙ (32|64), КОТОРЫЕ ЗАПУЩЕНЫ В СЕСИИ. PYTHON x64 ИМЕЕТ ВОЗМОЖНОСТЬ ВЗЗАИМОДЕЙСТВИЯ С x32 UIO ОБЪЕКТАМИ, НО МЫ РЕКОМЕНДУЕМ ДОПОЛНИТЕЛЬНО ИСПОЛЬЗОВАТЬ ИНТЕРПРЕТАТОР PYTHON x32 (ПОДРОБНЕЕ СМ. ФУНКЦИЮ Configure())

```
# UIDesktop: Взаимодействие с UI объектами приложений
from pyOpenRPA.Robot import UIDesktop
# 1C: UIO Селектор выбора базы
lDemoBaseUIOSelector = [{"title":"Запуск 1С:Предприятия","class_name":"V8TopLevelFrameTaxiStarter","ba
lBackendStr = UIDesktop.UIOSelector_SearchChildByMouse_UIOTree(lDemoBaseUIOSelector) # Получить список
```

Параметры:

- **inUIOSelector** (*list*, *обязательный*) – UIO селектор, который определяет UIO объект, для которого будет произведено извлечение всех атрибутов на всех уровнях.
- **inWaitBeforeSec** (*float*, *необязательный*) – Время ожидания перед началом поиска и перефокусировки. Данный аргумент может быть полезен для отображения полезной информации перед инициализацией режима поиска

Результат:

list, список атрибутов на каждом уровне UIO объекта

pyOpenRPA.Robot.UIDesktop.UIOSelector_SearchProcessNormalize_UIOSelector(*inControlSpecificationArray*)

[исходный код]

L-,W+: Нормализовать UIO селектор для дальнейшего использования в функциях поиска процесса, в котором находится искомый UIO объект. Если недопустимых атрибутов не присутствует, то оставить как есть.

```
# UIDesktop: Взаимодействие с UI объектами приложений
from pyOpenRPA.Robot import UIDesktop
# 1C: UIO Селектор выбора базы
lDemoBaseUIOSelectorDitry = [{"title":"Запуск 1С:Предприятия","class_name":"V8TopLevelFrameTaxiStarter
lDemoBaseUIOSelectorClean = UIDesktop.UIOSelector_SearchProcessNormalize_UIOSelector(lDemoBaseUIOSelec
```

Параметры:

inControlSpecificationArray (*list*, *обязательный*) – UIO селектор, который определяет UIO объект, для которого будет представлен перечень доступных активностей.

Результат:

нормализованный UIO селектор

pyOpenRPA.Robot.UIDesktop.UIOSelector_SearchUIONormalize_UIOSelector(*inControlSpecificationArray*)

[исходный код]

L-,W+: Нормализовать UIO селектор для дальнейшего использования в функциях поиск UIO объекта. Если недопустимых атрибутов не присутствует, то оставить как есть.

```
# UIDesktop: Взаимодействие с UI объектами приложений
from pyOpenRPA.Robot import UIDesktop
# 1C: UIO Селектор выбора базы
lDemoBaseUIOSelectorDitry = [{"title":"Запуск 1С:Предприятия","class_name":"V8TopLevelFrameTaxiStarter
lDemoBaseUIOSelectorClean = UIDesktop.UIOSelector_SearchUIONormalize_UIOSelector(lDemoBaseUIOSelectorD
```

Параметры:

inControlSpecificationArray (*list*, *обязательный*) – UIO селектор, который определяет UIO объект, для которого будет представлен перечень доступных активностей.

Результат:

нормализованный UIO селектор

pyOpenRPA.Robot.UIDesktop.UIOSelector_To_CSS(*inUIOSelector*) [исходный код]

L+,W+:Выполнить конвертацию селектора из UIO в CSS

```
# UIDesktop: Взаимодействие с UI объектами приложений
from pyOpenRPA.Robot import UIDesktop
lUIOSelector = [{"title":"Запуск 1С:Предприятия","class_name":"V8TopLevelFrameTaxiStarter","backend":"
lCSSSelector = UIDesktop.UIOSelector_To_CSS(inUIOSelector=lUIOSelector)
```

Параметры:

inUIOSelector (*list*, *обязательный*) – Селектор в формате UIO

pyOpenRPA.Robot.UIDesktop.UIOSelector_To_XPATH(*inUIOSelector*) [исходный код]

L+,W+:Выполнить конвертацию селектора из UIO в XPATH

```
# UIDesktop: Взаимодействие с UI объектами приложений
from pyOpenRPA.Robot import UIDesktop
lUIOSelector = [{"title":"Запуск 1С:Предприятия","class_name":"V8TopLevelFrameTaxiStarter","backend":"
lXPATHSelector = UIDesktop.UIOSelector_To_XPATH(inUIOSelector=lUIOSelector)
```

Параметры:

inUIOSelector (*list*, *обязательный*) – Селектор в формате UIO

pyOpenRPA.Robot.UIDesktop.UIOSelector_TryRestore_Dict(*inSpecificationList*) [исходный код]

L-,W+: Восстановить окно приложения на экране по UIO селектору inSpecificationList, если оно было свернуто. Функция обернута в try .. except - ошибок не возникнет.

!ВНИМАНИЕ! ДАННАЯ ФУНКЦИОНАЛЬНОСТЬ УЖЕ ИСПОЛЬЗУЕТСЯ В РЯДЕ ДРУГИХ ФУНКЦИЙ ТАК КАК АДРЕССАЦИЯ ПО UIA FRAMEWORK НЕДОСТУПНА, ЕСЛИ ПРИЛОЖЕНИЕ СВЕРНУТО.

!ВНИМАНИЕ! ДАННАЯ ФУНКЦИОНАЛЬНОСТЬ В АВТОМАТИЧЕСКОМ РЕЖИМЕ ПОДДЕРЖИВАЕТ ВСЕ РАЗРЯДНОСТИ ПРИЛОЖЕНИЙ (32|64), КОТОРЫЕ ЗАПУЩЕНЫ В СЕСИИ. PYTHON x64 ИМЕЕТ ВОЗМОЖНОСТЬ ВЗЗАИМОДЕЙСТВИЯ С x32 UIO ОБЪЕКТАМИ, НО МЫ РЕКОМЕНДУЕМ ДОПОЛНИТЕЛЬНО ИСПОЛЬЗОВАТЬ ИНТЕРПРЕТАТОР PYTHON x32 (ПОДРОБНЕЕ СМ. ФУНКЦИЮ Configure())

```
# UIDesktop: Взаимодействие с UI объектами приложений
from pyOpenRPA.Robot import UIDesktop
# 1C: UIO Селектор выбора базы
lDemoBaseUIOSelector = [{"title":"Запуск 1С:Предприятия","class_name":"V8TopLevelFrameTaxiStarter","ba
UIDesktop.UIOSelector_TryRestore_Dict(lDemoBaseUIOSelector) # Попытка восстановления свернутого окна п
```

Параметры:

inSpecificationList (*list*, *обязательный*) – UIO селектор, который определяет UIO объект, для которого будет произведено извлечение всех атрибутов на всех уровнях.

pyOpenRPA.Robot.UIDesktop.UIOSelectorsSecs_WaitAppear_List(*inSpecificationListList*, *inWaitSecs=86400.0*, *inFlagWaitAllInMoment=False*, *inFlagRaiseException=True*) [исходный код]

L-,W+: Ожидать появление хотя бы 1-го / всех UIO объектов по заданным UIO селекторам

!ВНИМАНИЕ! ДАННАЯ ФУНКЦИОНАЛЬНОСТЬ В АВТОМАТИЧЕСКОМ РЕЖИМЕ ПОДДЕРЖИВАЕТ ВСЕ РАЗРЯДНОСТИ ПРИЛОЖЕНИЙ (32|64), КОТОРЫЕ ЗАПУЩЕНЫ В СЕСИИ. PYTHON x64 ИМЕЕТ ВОЗМОЖНОСТЬ ВЗЗАИМОДЕЙСТВИЯ С x32 UIO ОБЪЕКТАМИ, НО МЫ РЕКОМЕНДУЕМ ДОПОЛНИТЕЛЬНО ИСПОЛЬЗОВАТЬ ИНТЕРПРЕТАТОР PYTHON x32 (ПОДРОБНЕЕ СМ. ФУНКЦИЮ Configure())

```
# UIDesktop: Взаимодействие с UI объектами приложений
from pyOpenRPA.Robot import UIDesktop
# 1C: UIO Селектор выбора базы
lDemoBaseUIOSelector = [{"title":"Запуск 1С:Предприятия","class_name":"V8TopLevelFrameTaxiStarter","ba
lNotepadOKSelector = [{"title":"notepad"},{"title":"OK"}]
lNotepadCancelSelector = [{"title":"notepad"},{"title":"Cancel"}]
lDemoBaseUIOExistList = UIDesktop.UIOSelectorsSecs_WaitAppear_List([lDemoBaseUIOSelector, lNotepadOKSe
```

Параметры:

- **inSpecificationListList** (*list*, *обязательный*) –

  Список UIO селекторов, которые определяют критерии поиска UIO объектов

  Пример: [[{«title»:»notepad»},{«title»:»OK»}], [{«title»:»notepad»},{«title»:»Cancel»}]

  ]

- **inWaitSecs** (*float*, *необязательный*) – Количество секунд, которые отвести на ожидание UIO объектов. По умолчанию 24 часа (86400 секунд)
- **inFlagRaiseException** (*bool*, *опциональный*) – True - формировать ошибку exception, если платформа не обнаружина ни одного UIO объекта по заданному UIO селектору. False - обратный случай (может привести к ошибочным результатам). По умолчанию True.
- **inFlagWaitAllInMoment** – True - Ожидать до того момента, пока не появятся все запрашиваемые UIO объекты на рабочей области

Результат:

Список индексов, которые указывают на номер входящих UIO селекторов, которые были обнаружены на рабочей области. Пример: [0,2]

---

pyOpenRPA.Robot.UIDesktop.UIOSelectorsSecs_WaitDisappear_List(*inSpecificationListList*, *inWaitSecs=86400.0*, *inFlagWaitAllInMoment=False*, *inFlagRaiseException=True*) [исходный код]

L-,W+: Ожидать исчезновение хотя бы 1-го / всех UIO объектов по заданным UIO селекторам

!ВНИМАНИЕ! ДАННАЯ ФУНКЦИОНАЛЬНОСТЬ В АВТОМАТИЧЕСКОМ РЕЖИМЕ ПОДДЕРЖИВАЕТ ВСЕ РАЗРЯДНОСТИ ПРИЛОЖЕНИЙ (32|64), КОТОРЫЕ ЗАПУЩЕНЫ В СЕСИИ. PYTHON x64 ИМЕЕТ ВОЗМОЖНОСТЬ ВЗАИМОДЕЙСТВИЯ С x32 UIO ОБЪЕКТАМИ, НО МЫ РЕКОМЕНДУЕМ ДОПОЛНИТЕЛЬНО ИСПОЛЬЗОВАТЬ ИНТЕРПРЕТАТОР PYTHON x32 (ПОДРОБНЕЕ СМ. ФУНКЦИЮ Configure())

```
# UIDesktop: Взаимодействие с UI объектами приложений
from pyOpenRPA.Robot import UIDesktop
# 1C: UIO Селектор выбора базы
lDemoBaseUIOSelector = [{"title":"Запуск 1С:Предприятия","class_name":"V8TopLevelFrameTaxiStarter","ba
lNotepadOKSelector = [{"title":"notepad"},{"title":"OK"}]
lNotepadCancelSelector = [{"title":"notepad"},{"title":"Cancel"}]
lDemoBaseUIOExistList = UIDesktop.UIOSelectorsSecs_WaitDisappear_List([lDemoBaseUIOSelector, lNotepadO
```

Параметры:

- **inSpecificationListList** (*list*, *обязательный*) –

  Список UIO селекторов, которые определяют критерии поиска UIO объектов

  Пример: [[{«title»:»notepad»},{«title»:»OK»}], [{«title»:»notepad»},{«title»:»Cancel»}]

  ]

- **inWaitSecs** (*float*, *необязательный*) – Количество секунд, которые отвести на ожидание исчезновения UIO объектов. По умолчанию 24 часа (86400 секунд)
- **inFlagRaiseException** (*bool*, *опциональный*) – True - формировать ошибку exception, если платформа не обнаружина ни одного UIO объекта по заданному UIO селектору. False - обратный случай (может привести к ошибочным результатам). По умолчанию True.
- **inFlagWaitAllInMoment** – True - Ожидать до того момента, пока не исчезнут все запрашиваемые UIO объекты на рабочей области

Результат:

Список индексов, которые указывают на номер входящих UIO селекторов, которые были обнаружены на рабочей области. Пример: [0,2]

---

pyOpenRPA.Robot.UIDesktop.UIOXY_SearchChild_ListDict(*inRootElement*, *inX*, *inY*, *inHierarchyList=None*, *inShowRootBool=False*) [исходный код]

L-,W+: Техническая функция: Получить иерархию вложенности UIO объекта по заданным корневому UIO объекту, координатам X и Y.

```
# UIDesktop: Взаимодействие с UI объектами приложений
from pyOpenRPA.Robot import UIDesktop
# 1C: UIO Селектор выбора базы
lDemoBaseUIOSelector = [{"title":"Запуск 1С:Предприятия","class_name":"V8TopLevelFrameTaxiStarter","ba
lUIO = UIDesktop.UIOSelector_Get_UIO(lDemoBaseUIOSelector) # Получить UIO объект с помощью UIO селекто
lUIOHierarchyList = UIDesktop.UIOXY_SearchChild_ListDict(lUIO, 100, 200) # Получить UIO объект с помощ
```

Параметры:

- **inRootElement** (*object UIO*, *обязательный*) – родительский UIO объект, полученный ранее с помощью UIO селектора.
- **inX** (*int*, *обязательный*) – родительский UIO объект, полученный ранее с помощью UIO селектора.
- **inY** (*int*, *обязательный*) – родительский UIO объект, полученный ранее с помощью UIO селектора.
- **inShowRootBool** (*bool*, *необязательный*) – True - в результирующем списке показывать корневой объект, от которого был инициирован вызов. False - не показывать корневой объект

Результат:

Список словарей - уровней UIO объектов

pyOpenRPA.Robot.UIDesktop.UIO_FocusHighlight(*lWrapperObject, colour='green', thickness=2, fill=None, rect=None*) [исходный код]

L-,W+: Установить фокус и выполнить подсветку UIO объекта на экране

```
# UIDesktop: Взаимодействие с UI объектами приложений
from pyOpenRPA.Robot import UIDesktop
# 1C: UIO Селектор выбора базы
lDemoBaseUIOSelector = [{"title":"Запуск 1С:Предприятия","class_name":"V8TopLevelFrameTaxiStarter","ba
lUIO = UIDesktop.UIOSelector_Get_UIO(lDemoBaseUIOSelector) # Получить UIO объект по UIO селектору
UIDesktop.UIO_FocusHighlight(lUIO) # Установить фокус и подсветить UIO объект по UIO селектору зеленым
```

Параметры:

- **lWrapperObject** (*object UIO, обязательный*) – UIO объект, который будет подсвечен
- **colour** (*str, необязательный*) – цвет подсветки UIO объекта. Варианты: „red“, „green“, „blue“. По умолчанию „green“
- **thickness** (*int, необязательный*) – толщина подсветки UIO объекта. По умолчанию 2

pyOpenRPA.Robot.UIDesktop.UIO_GetCtrlIndex_Int(*inElement*) [исходный код]

L-,W+: Получить индекс UIO объекта inElement в списке родительского UIO объекта.

```
# UIDesktop: Взаимодействие с UI объектами приложений
from pyOpenRPA.Robot import UIDesktop
# 1C: UIO Селектор выбора базы
lDemoBaseUIOSelector = [{"title":"Запуск 1С:Предприятия","class_name":"V8TopLevelFrameTaxiStarter","ba
lUIO = UIDesktop.UIOSelector_Get_UIO(lDemoBaseUIOSelector) # Получить UIO объект по UIO селектору.
lUIOIndexInt = UIDesktop.UIO_GetCtrlIndex_Int(lUIO) # Получить индекс UIO объекта в списке у родительс
```

Параметры:

inElement (*list, обязательный*) – UIO объект, для которого требуется определить индекс в списке родительского UIO объекта.

Результат:

int, индекс UIO объекта в списке родительского UIO объекта

pyOpenRPA.Robot.UIDesktop.UIO_GetValue_Str(*inUIO*) [исходный код]

Получить значение UI объекта по методу get_value. Если нет такого метода или ошибка выполнения - вернуть None

```
# UIDesktop: Взаимодействие с UI объектами приложений
from pyOpenRPA.Robot import UIDesktop
lFilterStr = "1C*"
lFilterType = "WC"
UIOList = UIDesktop.UIO_Search_UIOTree(inFilterStr=lFilterStr, inFilterType=lFilterType)
lValueStr = UIO_GetValue_Str(inUIO)
```

Параметры:

inUIO (*UIO*) – UI объект

Результат:

Значение UI объекта

Тип результата:

str or None

pyOpenRPA.Robot.UIDesktop.UIO_Highlight(*lWrapperObject, colour='green', thickness=2, fill=None, rect=None, inFlagSetFocus=False*) [исходный код]

L-,W+: Выполнить подсветку UIO объекта на экране

```
# UIDesktop: Взаимодействие с UI объектами приложений
from pyOpenRPA.Robot import UIDesktop
# 1C: UIO Селектор выбора базы
lDemoBaseUIOSelector = [{"title":"Запуск 1С:Предприятия","class_name":"V8TopLevelFrameTaxiStarter","ba
lUIO = UIDesktop.UIOSelector_Get_UIO(lDemoBaseUIOSelector) # Получить UIO объект по UIO селектору
UIDesktop.UIO_Highlight(lUIO) # Подсветить UIO объект по UIO селектору зеленым цветом с толщиной подсв
```

Параметры:

- **lWrapperObject** (*object UIO, обязательный*) – UIO объект, который будет подсвечен
- **colour** (*str, необязательный*) – цвет подсветки UIO объекта. Варианты: „red“, „green“, „blue“. По умолчанию „green“
- **thickness** (*int, необязательный*) – толщина подсветки UIO объекта. По умолчанию 2
- **inFlagSetFocus** (*bool, необязательный*) – признак установки фокуса на UIO объект перед подсветкой. По умолчанию False

pyOpenRPA.Robot.UIDesktop.UIO_Search_UIOList(*inFilterStr, inFilterType, inParentUIOSelector=None, inParentUIO=None, inFlagRaiseException=True, inRuntimeLimit=60, inTimeStartPoint=None*) [исходный код]

L-,W+: Получить список UIO объектов по заданной строке. Поиск производится по всем атрибутам UIO объектов строкового типа.

Параметры:

- **inFilterStr** (*str, обязательный*) – Строка, по которой производится поиск
- **inFilterType** (*str, обязательный*) – Тип задаваемой для поиска строки. Доступно: STR (str), RE (regexp) и WC (wildcard)
- **inParentUIOSelector** (*list, опциональный*) – UIO Селектор, который определяет корень, от которого производить поиск. По умолчанию None (поиск UIO объектов среди всех доступных приложений windows, которые запущены на текущей сессии)
- **inParentUIO** (*UIO объект, опциональный*) – Родительский элемент, от которого выполнить поиск UIO объектов по заданной строке. По умолчанию None (поиск UIO объектов среди всех доступных приложений windows, которые запущены на текущей сессии)
- **inFlagRaiseException** (*bool, опциональный*) – True - формировать ошибку exception, если платформа не обнаружина ни одного UIO объекта по заданному UIO селектору. False - обратный случай (может привести к ошибочным результатам). По умолчанию True.
- **inRuntimeLimit** (*float, опциональный*) – Лимит времени, выделенный на поиск UIO объектов. По умолчанию - 60 (в сек.)
- **inTimeStartPoint** (*float, опциональный*) – Точка начала отсчета для контроля лимита времени, выделенного на поиск UIO объектов. По умолчанию None (самоопределяется при запуске функции)

Результат:

Список UIO объектов, которые удовлетворяют заданной строке

pyOpenRPA.Robot.UIDesktop.UIO_Search_UIOTree(*inFilterStr*, *inFilterType*, *inParentUIOSelector=None*, *inParentUIOList=None*, *inBackendStr='win32'*) [исходный код]

L-,W+: Получить список UIO объектов по заданной строке. Поиск производится по всем атрибутам UIO объектов строкового типа.

ОБНОВЛЕНИЕ 1.4.0: Функция обновлена под использование в новой версии студии

!ВНИМАНИЕ! ДАННАЯ ФУНКЦИОНАЛЬНОСТЬ В АВТОМАТИЧЕСКОМ РЕЖИМЕ ПОДДЕРЖИВАЕТ ВСЕ РАЗРЯДНОСТИ ПРИЛОЖЕНИЙ (32|64), КОТОРЫЕ ЗАПУЩЕНЫ В СЕСИИ. PYTHON x64 ИМЕЕТ ВОЗМОЖНОСТЬ ВЗЗАИМОДЕЙСТВИЯ С x32 UIO ОБЪЕКТАМИ, НО МЫ РЕКОМЕНДУЕМ ДОПОЛНИТЕЛЬНО ИСПОЛЬЗОВАТЬ ИНТЕРПРЕТАТОР PYTHON x32 (ПОДРОБНЕЕ СМ. ФУНКЦИЮ Configure())

```
# UIDesktop: Взаимодействие с UI объектами приложений
from pyOpenRPA.Robot import UIDesktop
lFilterStr = "1C*"
lFilterType = "WC"
UIOList = UIDesktop.UIO_Search_UIOTree(inFilterStr=lFilterStr, inFilterType=lFilterType)
```

Параметры:

- **inFilterStr** (*str, обязательный*) – Строка, по которой производится поиск
- **inFilterType** (*str, обязательный*) – Тип задаваемой для поиска строки. Доступно: STR (str), RE (regexp) и WC (wildcard)

Результат:

list, список атрибутов на каждом уровне UIO объекта

pyOpenRPA.Robot.UIDesktop.XPATH_To_UIOSelector(*inXPATHSelector*) [исходный код]

L+,W+:Выполнить конвертацию селектора из XPATH в UIO

```
# UIDesktop: Взаимодействие с UI объектами приложений
from pyOpenRPA.Robot import UIDesktop
lXPATHSelector = "/Запуск 1С:Предприятия[@class_name='V8TopLevelFrameTaxiStarter'][@backend='uia']"
lUIOSelector = UIDesktop.XPATH_To_UIOSelector(inXPATHSelector=lXPATHSelector)
```

Параметры:

inUIOSelector (*str, обязательный*) – Селектор в формате XPATH

Селектор UIO

Селектор UIO - адрес одного и/или более UIO объектов. Селектор представлен в формате списка (list) словарей (dict) атрибутивных критериев. Поддерживает формат JSON, что позволяет обеспечить удобство форматирования и передачи через web интерфейс студии / оркестратора.

UIO селектор — это список характеристических словарей (спецификаций UIO). Данные спецификации UIO содержат условия, с помощью которых библиотека pyOpenRPA определит UIO, удовлетворяющий условиям, заданным в спецификации UIO. Индекс спецификации UIO в списке UIO селектора характеризует уровень вложенности целевого UIO. Говоря другим языком, UIO селектор — это перечень условий, под которые может попасть 0, 1 или n UIO.

Ниже приведен перечень атрибутов — условий, которые можно использовать в спецификациях UIO:

Формат селектора:

[

{

«depth_start» :: [int, начинается с 1] :: глубина, с которой начинается поиск (по умолчанию 1),

«depth_end» :: [int, начинается с 1] :: глубина, до которой ведется поиск (по умолчанию 1),

«ctrl_index» || «index» :: [int, начинается с 0] :: индекс UIO в списке у родительского UIO,

«title» :: [str] :: идентичное наименование атрибута *title* искомого объекта UIO,

«title_re» :: [str] :: регулярное выражение (python диалект) для отбора UIO, у которого атрибут *title* должен удовлетворять условию данного регулярного выражения,

«rich_text» :: [str] :: идентичное наименование атрибута *rich_text* искомого объекта UIO,

«rich_text_re» :: [str] :: регулярное выражение (python диалект) для отбора UIO, у которого атрибут *rich_text* должен удовлетворять условию данного регулярного выражения,

«class_name» :: [str] :: идентичное наименование атрибута *class_name* искомого объекта UIO,

«class_name_re» :: [str] :: регулярное выражение (python диалект) для отбора UIO, у которого атрибут *class_name* должен удовлетворять условию данного регулярного выражения,

«friendly_class_name» :: [str] :: идентичное наименование атрибута *friendly_class_name* искомого объекта UIO,

«friendly_class_name_re» :: [str] :: регулярное выражение (python диалект) для отбора UIO, у которого атрибут *friendly_class_name* должен удовлетворять условию данного регулярного выражения,

«control_type» :: [str] :: идентичное наименование атрибута *control_type* искомого объекта UIO,

«control_type_re» :: [str] :: регулярное выражение (python диалект) для отбора UIO, у которого атрибут *control_type* должен удовлетворять условию данного регулярного выражения,

«is_enabled» :: [bool] :: признак, что UIO доступен для выполнения действий,

«is_visible» :: [bool] :: признак, что UIO отображается на экране,

«backend» :: [str, «win32» || «uia»] :: вид способа адресации к UIO (по умолчанию «win32»). Внимание! Данный атрибут может быть указан только для первого элемента списка UIO селектора. Для остальных элементов списка данный атрибут будет проигнорирован.

}, { … спецификация UIO следующего уровня иерархии }

]

Пример UIO селектора: [

{«class_name»:»CalcFrame», «backend»:»win32»}, # Спецификация UIO 1-го уровня вложенности {«title»:»Hex», «depth_start»:3, «depth_end»: 3} # Спецификация UIO 1+3-го уровня вложенности (так как установлены атрибуты depth_start|depth_stop, определяющие глубину поиска UIO)

]

UIO объект - свойства и методы (общие)

- process_id(): Возвращает идентификатор процесса, которому принадлежит это окно
- window_text(): Текст окна элемента. Довольно много элементов управления имеют другой текст, который виден, например, элементы управления редактированием обычно имеют пустую строку для window_text, но все равно имеют текст, отображаемый в окне редактирования.
- rectangle(): Возвращает прямоугольник элемента: {«сверху», «слева», «справа», «снизу»} Прямоугольник() - это прямоугольник элемента на экране. Координаты указаны в левом верхнем углу экрана. Этот метод возвращает прямоугольную структуру, которая имеет атрибуты - top, left, right, bottom. и имеет методы width() и height(). См. раздел win32structures.Прямую кишку для получения дополнительной информации.
- right_click_input(coords=(None, None)): Щелкните правой кнопкой мыши на указанных координатах
- click_input(button="left", coords=(None, None), button_down=True, button_up=True, double=False, wheel_dist=0, use_log=True, pressed="", absolute=False, key_down=True, key_up=True): Щелкните по указанным координатам кнопкой мыши, чтобы щелкнуть. Один из «влево», «вправо», «посередине» или «x» (по умолчанию: «влево», «переместить» - это особый случай) определяет координаты, по которым нужно щелкнуть.(По умолчанию: центр элемента управления) дважды Укажите, следует ли выполнять двойной щелчок или нет (по умолчанию: False) wheel_dist Расстояние для перемещения колеса мыши (по умолчанию: 0) Внимание: Этот метод отличается

от метода щелчка тем, что он требует, чтобы элемент управления был виден на экране, но выполняет более реалистичную симуляцию щелчка. Этот метод также уязвим, если пользователь перемещает мышь, поскольку это может легко переместить мышь с элемента управления до завершения click_input.

- double_click_input(button="left", coords=(None, None)): Дважды щелкните по указанным координатам
- press_mouse_input(button="left", coords=(None, None), pressed="", absolute=True, key_down=True, key_up=True): Нажмите кнопку мыши с помощью SendInput
- drag_mouse_input(dst=(0, 0), src=None, button="left", pressed="", absolute=True): Нажмите на src, перетащите его и перетащите на dst dst - это объект-оболочка назначения или просто координаты. src - это исходный объект-оболочка или координаты. Если src равен None, self используется в качестве исходного объекта. кнопка - это кнопка мыши, которую нужно удерживать во время перетаскивания. Это может быть "влево", "вправо", "посередине" или "x". Нажата клавиша на клавиатуре, которую нужно нажимать во время перетаскивания. абсолютные указывает, следует ли использовать абсолютные координаты для расположения указателя мыши
- wheel_mouse_input(coords=(None, None), wheel_dist=1, pressed=""): Прокрутить колесо мыши
- draw_outline(colour="green", thickness=2, fill=<MagicMock name="mock.win32defines.BS_NULL" id="140124673757368">, rect=None): Нарисуйте контур вокруг окна. цвет может быть либо целым числом, либо одним из «красного», «зеленого», «синего» (по умолчанию «зеленый») толщина толщина прямоугольника (по умолчанию 2) заливка как заполнить прямоугольник (по умолчанию BS_NULL) укажите координаты прямоугольника для рисования (по умолчанию используется прямоугольник элемента управления)
- element_info: Свойство, доступное только для чтения, для получения объекта ElementInfo
- from_point(x, y): Получить объект-оболочку для элемента в заданных координатах экрана (x, y)
- get_properties(): Возвращает свойства элемента управления в виде словаря.
- is_child(parent): Возвращает значение True, если этот элемент является дочерним элементом 'parent'. Элемент является дочерним элементом другого элемента, когда он является прямым элементом другого элемента. Элемент является прямым потомком данного элемента, если родительский элемент является цепочкой родительских элементов для дочернего элемента.
- is_dialog(): Возвращает значение True, если элемент управления является окном верхнего уровня
- is_enabled(): Независимо от того, включен элемент или нет. Проверяет, что как родительский элемент верхнего уровня (возможно, диалоговое окно), которому принадлежит этот элемент, так и сам элемент включены.Если вы хотите дождаться, пока элемент станет включенным (или дождаться, пока он станет отключенным), используйте Application.wait(„visible") или Application.wait_not(„visible").Если вы хотите немедленно вызвать исключение, если элемент не включен, вы можете использовать BaseWrapper.verify_enabled(). Функция BaseWrapper.Verify Ready() вызывается, если окно одновременно не видно и не включено.
- is_visible(): Является ли элемент видимым или нет. Проверяет, видны ли как родительский элемент верхнего уровня (возможно, диалоговое окно), которому принадлежит этот элемент, так и сам элемент. Если вы хотите дождаться, пока элемент станет видимым (или дождаться, пока он станет скрытым), используйте Application.wait(„visible") или Application.wait_not(„visible"). Если вы хотите немедленно вызвать исключение, если элемент не виден, вы можете использовать BaseWrapper.verify_visible(). Базовая оболочка.verify_actible() вызывается, если элемент одновременно не виден и не включен.
- parent(): Возвращает родительский элемент этого элемента Обратите внимание, что родительским элементом элемента управления не обязательно является диалоговое окно или другое главное окно. Например, поле группы может быть родительским для некоторых переключателей. Чтобы получить главное (или окно верхнего уровня), затем используйте BaseWrapper.top_level_parent().
- root(): Возвращаемая оболочка для корневого элемента (рабочий стол)
- set_focus(): Установить фокус на этот элемент
- texts(): Возвращает текст для каждого элемента этого элемента управления Это список строк для элемента управления. Часто переопределяется извлечение всех строк из элемента управления с несколькими элементами. Это всегда список с одной или несколькими строками: Первый элемент - это текст окна элемента управления Последующие элементы содержат текст любых элементов элемента управления (например, элементы в listbox/combobox, вкладки в tabcontrol)
- type_keys(keys, pause=None, with_spaces=False, with_tabs=False, with_newlines=False, turn_off_numlock=True, set_foreground=True, vk_packet=True): Введите ключи для элемента с помощью клавиатуры.send_keys. Ограниченная функциональность. Для более полной функциональности рекомендуем ознакомится с pyOpenPRA.Robot.Keyboard
- was_maximized(): Проверить, было ли окно развернуто перед сворачиванием или нет

UIO свойства и методы (дополнение к базовым методам для win32 элементов)

Кнопка (Button || CheckBox || RadioButton || GroupBox)

- check(): Установить флажок
- uncheck(): Снять флажок
- get_check_state(): Вернуть состояние проверки флажка. Состояние проверки представлено целым числом 0 - непроверено 1 - проверено 2 - неопределенно. Следующие константы определены в

модуле win32defines BST_UNCHECKED = 0 BST_CHECKED = 1 BST_INDETERMINATE = 2
- click(button="left", pressed="", coords=(0, 0), double=False, absolute=False): Клик на кнопку управления
- is_checked(): Возвращает True, если флажок установлен, False, если флажок не установлен, None, если значение не определено
- is_dialog(): Кнопки никогда не являются диалоговыми окнами, поэтому возвращайте значение False
- set_check_indeterminate(): Установить флажок в положение неопределенный
- friendly_class_name(): Возвращает имя класса кнопки. Они могут выглядеть следующим образом: Кнопки, этот метод возвращает "Button"; Флажки, этот метод возвращает "флажок"; RadioButtons, этот метод возвращает "RadioButton"; GroupBoxes, этот метод возвращает "GroupBox"

Поле выбора нескольких значений из списка (ComboBox)

- friendlyclassname = „ComboBox"
- windowclasses = [„ComboBox", „WindowsForms\d*.COMBOBOX..*", „.*ComboBox"]
- dropped_rect(): Получить выпадающий прямоугольник в поле со списком
- get_properties(): Возвращает свойства элемента управления в виде словаря
- item_count(): Возвращает количество элементов в поле со списком
- item_data(item): Возвращает данные элемента, связанные с элементом, если таковые имеются
- item_texts(): Возвращает текст элементов выпадающего списка
- select(item): Выбрать элемент со списком элемент может быть либо индексом элемента для выбора на основе 0, либо строкой, которую вы хотите выбрать
- selected_index(): Возвращает выбранный индекс
- selected_text():Возвращает выделенный текст
- texts(): Возвращает текст элементов в выпадающем списке

Поле ввода (Edit)

- friendlyclassname = „Edit"
- windowclasses = [„Edit", „.*Edit", „TMemo", „WindowsForms\d*.EDIT..*", „ThunderTextBox", „ThunderRT6TextBox"]
- get_line(line_index): Возвращает указанную строку
- line_count(): Возвращает, сколько строк есть в редактировании
- line_length(line_index): Возвращает количество символов в строке
- select(start=0, end=None): Установите выбор редактирования элемента управления редактированием
- selection_indices(): Начальный и конечный индексы текущего выбора
- set_edit_text(text, pos_start=None, pos_end=None): Задать текст элемента управления редактированием
- set_text(text, pos_start=None, pos_end=None): Задать текст элемента управления редактированием
- set_window_text(text, append=False): Переопределите set_window_text для элементов управления редактированием, поскольку он не должен использоваться для элементов управления редактированием. Элементы управления редактированием должны использовать либо set_edit_text(), либо type_keys() для изменения содержимого элемента управления редактированием.
- text_block(): Получить текст элемента управления редактированием
- texts(): Получить текст элемента управления редактированием

Поле выбора 1-го значения из списка (ListBox)

- friendlyclassname = „ListBox"
- windowclasses = [„ListBox", „WindowsForms\d*.LISTBOX..*", „.*ListBox"]
- get_item_focus(): Возвращает индекс текущего выбора в списке
- is_single_selection(): Проверить, имеет ли поле списка режим одиночного выбора
- item_count(): Возвращает количество элементов в списке
- item_data(i): Возвращает item_data, если таковые имеются, связанные с элементом
- item_texts(): Возвращает текст элементов списка
- select(item, select=True): Выбрать элемент списка элемент может быть либо индексом элемента для выбора на основе 0, либо строкой, которую вы хотите выбрать
- selected_indices(): Выбранные в данный момент индексы списка
- set_item_focus(item): Установить фокус по элементу
- texts(): Получить текст элемента управления редактированием

Выпадающее меню (PopupMenu)

- friendlyclassname = „PopupMenu"
- windowclasses = [„#32768"]
- is_dialog(): Возвращает, является ли это диалогом

Текст (Static)

- friendlyclassname= „Static“
- windowclasses= [„Static“, „WindowsForms\d*.STATIC..*“, „TPanel“, „.*StaticText“]

Инициализация 2-х разрядностей для UIO

pyOpenRPA позволяет обеспечить максимальную совместимость со всеми приложениями, которые выполняются на компьютере. Мы рекомендуем разрабатывать робота под интерпретатором Python x64. В дополнение к нему Вы можете подключить Python x32 (см. ниже пример подключения). Если планируемый робот не будет взаимодействовать через pyOpenRPA.Robot.UIDesktop с другой разрядность, то эту настройку можно не применять.

```
from pyOpenRPA.Robot import UIDesktop
# В нашем случае процесс робота будет исполняться на Python x64. Дополнительно подключим Python x32 (дела
lPyOpenRPA_SettingsDict = {
        "Python32FullPath": "..\\Resources\\WPy32-3720\\python-3.7.2\\python.exe",# Путь к интерпретатору
        "Python64FullPath": "..\\Resources\\WPy64-3720\\python-3.7.2.amd64\\python.exe", # Путь к интерпр
        "Python32ProcessName": "pyOpenRPA_UIDesktopX32.exe", # Наименование процесса робота x32 в диспетч
        "Python64ProcessName": "pyOpenRPA_UIDesktopX64.exe" # Наименование процесса робота x64 в диспетче
}
# Инициализировать 2-й разрядность.
UIDesktop.Utils.ProcessBitness.SettingsInit(lPyOpenRPA_SettingsDict)
# Теперь при вызове функций pyOpenRPA.Robot.UIDesktop платформа pyOpenRPA будет отслеживать разрядность п
```

Быстрая навигация

- Сообщество pyOpenRPA (telegram)
- Сообщество pyOpenRPA (tenchat)
- Сообщество pyOpenRPA (вконтакте)
- Презентация pyOpenRPA
- Портал pyOpenRPA
- Репозиторий pyOpenRPA

.. v1.4.0 replace:: v1.4.0

Предыдущая Следующая

3. Функции UIWeb

Общее

Здесь представлено описание всех функций, необходимых для максимально эффективного управления web интерфейсами различных приложений.

Описание функций

Описание каждой функции начинается с обозначения L+,W+, что означает, что функция поддерживается в ОС Linux (L) и Windows (W)

Functions:

`BrowserChange` (inBrowser)	L+,W+: Выполнить смену активного браузера (при необходим
`BrowserChromeStart` ([inDriverExePathStr, ...])	L+,W+: Выполнить запуск браузера Chrome.
`BrowserClose` ()	L+,W+: Закрыть браузер
`BrowserFocus` ()	L+,W+: Выполнить фокусировку на окно браузера
`MouseSearchChild` ()	L+,W+: Инициировать визуальный поиск UIO объекта с помо
`MouseSearchChildTree` ([inWaitBeforeSec])	L-,W+: Получить список уровней UIO объекта с указнием все
`PageClose` ([inIndexInt])	L+,W+: Закрыть текущую вкладку или вкладку, расположенн
`PageCount` ()	L+,W+: Вернуть количество открытых вкладок в браузере
`PageJSExecute` (inJSStr, *inArgList)	L+,W+: Отправить на выполнение на сторону браузера код Ja
`PageNew` ([inURLStr])	L+,W+: Открыть новую вкладку в браузере
`PageOpen` (inURLStr)	L+,W+: Открыть страницу inURLStr в браузере и дождаться е
`PagePrint` ()	L+,W+: Открыть окно печати браузера.
`PageScrollTo` ([inVerticalPxInt, ...])	L+,W+: Выполнить прокрутку страницы (по вертикали или по
`PageSwitch` (inTabIndexInt)	L+,W+: Переключить вкладку по индексу inTabIndexInt
`SelectorConvert` (inSelector, inToTypeStr)	L+,W+: Перевести селектор в заданный тип.
`UIOAttributeDictGet` (inUIO)	L+,W+: Получить словарь атрибутов UI объекта.
`UIOAttributeGet` (inUIO, inAttributeStr)	L+,W+: Получить обычный (нестилевой) атрибут у UI элемент
`UIOAttributeRemove` (inUIO, inAttributeStr)	L+,W+: Удалить обычный (нестилевой) атрибут у UI элемента
`UIOAttributeSet` (inUIO, inAttributeStr, inValue)	L+,W+: Установить обычный (нестилевой) атрибут у UI элеме
`UIOAttributeStyleGet` (inUIO, inAttributeStr)	L+,W+: Получить стилевой атрибут у UI элемента.
`UIOAttributeStyleRemove` (inUIO, inAttributeStr)	L+,W+: Удалить стилевой атрибут у UI элемента.
`UIOAttributeStyleSet` (inUIO, inAttributeStr, ...)	L+,W+: Установить стилевой атрибут у UI элемента.

`UIOChildListAttributeDictGet` (inUIO)	L+,W+: Получить список словарь атрибутов для детей UI объ
`UIOClick` (inUIO)	L+,W+: Выполнить нажатие по элементу inUIO.
`UIOHighlight` (inUIO[, inIsFirst, ...])	L+,W+: Выполнить подсвечивание UI элемента с UIO.
`UIOMouseSearchInit` ()	L+,W+: Инициализирует процесс поиска UI элемента с помо
`UIOMouseSearchReturn` ([inStopSearchBool])	L+,W+: Возвращает UIO объект, над которым находится указ
`UIOSelectorActivityArgGet` (inUIOSelector, ...)	L+,W+: Сформировать преднастроенный список/словарь арг
`UIOSelectorActivityListGet` (inUIOSelector)	L+,W+: Получить список доступных действий/функций по UI
`UIOSelectorActivityRun` (inUIOSelector, ...[, ...])	L+,W+: Выполнить активность inActionName над UIO объекто
`UIOSelectorChildListAttributeDictGet` ([...])	L+,W+: Получить список словарей атриботов детей UI объек
`UIOSelectorClick` (inUIOSelectorStr)	L+,W+: Выполнить нажатие по элементу с селектором inUIO
`UIOSelectorDetect` (inUIOSelectorStr)	L+,W+: Идентифицировать стиль селектора (CSS или XPATH
`UIOSelectorFirst` ([inUIOSelectorStr, inUIO])	L+,W+: Получить UIO объект по UIO селектору.
`UIOSelectorFocusHighlight` (inUIOSelectorStr)	L+,W+: Выполнить подсвечивание UI элемента с селектором
`UIOSelectorHighlight` (inUIOSelectorStr[, ...])	L+,W+: Выполнить подсвечивание UI элемента с селектором
`UIOSelectorLevelInfoList` (inUIOSelector)	L+,W+: Получить список свойств всех уровней до UI объекта,
`UIOSelectorList` ([inUIOSelectorStr, inUIO])	L+,W+: Получить список UIO объектов по UIO селектору.
`UIOSelectorListAttributeDictGet` ([inUIOSelector])	L+,W+: Получить словарь атрибутов для UI объектов, которь
`UIOSelectorSetValue` (inUIOSelectorStr, inValue)	L+,W+: Установить значение элемента с селектором inUIOSel
`UIOSelectorWaitAppear` (inUIOSelectorStr[, ...])	L+,W+: Ожидать появление UI элемента на веб странице (бло
`UIOSelectorWaitDisappear` (inUIOSelectorStr[, ...])	L+,W+: Ожидать исчезновение UI элемента с веб страницы (6
`UIOSelector_To_XPATH` ([inUIOSelector])	L+,W+: Выполнить конвертацию селектора из UIO в XPATH
`UIOTextGet` (inUIO)	L+,W+: Получить текст UI элемента.
`XPATH_To_UIOSelector` ([inXPATHSelector])	L+,W+: Выполнить конвертацию селектора из XPATH в UIO

pyOpenRPA.Robot.UIWeb.BrowserChange(*inBrowser*) [исходный код]

L+,W+: Выполнить смену активного браузера (при необходимости).

```
# UIWeb: Взаимодействие с ui web
from pyOpenRPA.Robot import UIWeb
lBrowser1 = UIWeb.BrowserChromeStart()
UIWeb.BrowserChange(inBrowser=None)
lBrowser2 = UIWeb.BrowserChromeStart()
UIWeb.BrowserClose()
UIWeb.BrowserChange(inBrowser=lBrowser1)
UIWeb.BrowserClose()
```

Параметры:

inBrowser (*webdriver.Chrome*) – Объект браузера

pyOpenRPA.Robot.UIWeb.BrowserChromeStart(*inDriverExePathStr: Optional[str] = None, inChromeExePathStr: Optional[str] = None, inExtensionPathList: Optional[list] = None, inProfilePathStr: Optional[str] = None, inSaveAsPDFBool=False, inSavefileDefaultDirStr: Optional[str] = None, inUrlStr: Optional[str] = None, inModeStr: Optional[str] = None, inMaximizeBool: Optional[bool] = None*)→ WebDriver [исходный код]

L+,W+: Выполнить запуск браузера Chrome. Если вы скачали pyOpenRPA вместе с репозиторием, то будет использоваться встроенный браузер Google Chrome. Если установка pyOpenRPA производилась другим способом, то требуется указать расположение браузера Google Chrome и соответствующего WebDriver

```
# UIWeb: Взаимодействие с ui web
from pyOpenRPA.Robot import UIWeb
UIWeb.BrowserChromeStart()
UIWeb.BrowserClose()
```

Параметры:

inDriverExePathStr (*str, опционально*) – Путь до компонента webdriver.exe, по умолчанию None (путь до webdriver.exe, который расположен в репозитории pyOpenRPA)

:param inChromeExePathStr:Путь до компонента chrome.exe, по умолчанию None (путь до chrome.exe, который расположен в репозитории pyOpenRPA) :type inChromeExePathStr: str, опционально :param inExtensionPathList: Список путей, по которым располагаются расширения Chrome, по умолчанию None :type inExtensionPathList: list, опционально :param inProfilePathStr: Путь, по которому выполнить сохранения профиля Chrome (история, куки и т.д.), по умолчанию None (профиль не сохраняется) :type inProfilePathStr: str, опционально :param inSaveAsPDFBool: Флаг, который обеспечивает настройки окна печати вэб-страницы как «Сохранить как PDF», по умолчанию False (настройки по умолчанию) :type inSaveAsPDFBool: bool, опционально :param inSavefileDefaultDirStr: Путь, по которому выполнить сохранения файла (после работы с окном печать вэб-страницы браузера) (история, куки и т.д.), по умолчанию None (файл не сохраняется) :type inSavefileDefaultDirStr: str, опционально :param inUrlStr: Путь к странице, которую требуется открыть. По умолчанию адреса нет :type inUrlStr: str, опционально :param inModeStr: Формат запуска браузера. Доступные варианты: «NORMAL», «APP», «KIOSK». По умолчанию «NORMAL» :type inModeStr: str, опционально :param inMaximizeBool: True - развернуть на весь экран. False (по умолчанию) - не разворачивать. :type inMaximizeBool: bool, опционально :return: Объект браузера Google Chrome :rtype: webdriver.Chrome

pyOpenRPA.Robot.UIWeb.BrowserClose() [исходный код]

L+,W+: Закрыть браузер

```
# UIWeb: Взаимодействие с ui web
from pyOpenRPA.Robot import UIWeb
UIWeb.BrowserChromeStart()
UIWeb.PageOpen("https://mail.ru")
UIWeb.BrowserClose()
```

pyOpenRPA.Robot.UIWeb.BrowserFocus() [исходный код]

L+,W+: Выполнить фокусировку на окно браузера

```
# UIWeb: Взаимодействие с ui web
from pyOpenRPA.Robot import UIWeb
UIWeb.BrowserChromeStart()
UIWeb.BrowserFocus()
```

pyOpenRPA.Robot.UIWeb.MouseSearchChild() [исходный код]

L+,W+: Инициировать визуальный поиск UIO объекта с помощью указателя мыши. При наведении указателя мыши UIO объект выделяется зеленой рамкой. Остановить режим поиска можно с помощью зажима клавиши ctrl left на протяжении нескольких секунд. После этого в веб окне студии будет отображено дерево расположения искомого UIO объекта.

```
# UIWeb: Взаимодействие с ui web
from pyOpenRPA.Robot import UIWeb
import time
UIWeb.BrowserChromeStart()
UIWeb.PageOpen("https://mail.ru")
lUIO = MouseSearchChild()
UIWeb.BrowserClose()
```

Результат:

UIO объект или None (если UIO не был обнаружен)

pyOpenRPA.Robot.UIWeb.MouseSearchChildTree(*inWaitBeforeSec=0.0*) [исходный код]

L-,W+: Получить список уровней UIO объекта с указнием всех имеющихся атрибутов.

!ВНИМАНИЕ! ДАННАЯ ФУНКЦИОНАЛЬНОСТЬ В АВТОМАТИЧЕСКОМ РЕЖИМЕ ПОДДЕРЖИВАЕТ ВСЕ РАЗРЯДНОСТИ ПРИЛОЖЕНИЙ (32|64), КОТОРЫЕ ЗАПУЩЕНЫ В СЕСИИ. PYTHON x64 ИМЕЕТ ВОЗМОЖНОСТЬ ВЗЗАИМОДЕЙСТВИЯ С x32 UIO ОБЪЕКТАМИ, НО МЫ РЕКОМЕНДУЕМ ДОПОЛНИТЕЛЬНО ИСПОЛЬЗОВАТЬ ИНТЕРПРЕТАТОР PYTHON x32 (ПОДРОБНЕЕ СМ. ФУНКЦИЮ Configure())

```
# UIWeb: Взаимодействие с ui web
from pyOpenRPA.Robot import UIWeb
import time
UIWeb.BrowserChromeStart()
UIWeb.PageOpen("https://mail.ru")
lUIO = MouseSearchChildTree()
UIWeb.BrowserClose()
```

Параметры:

inWaitBeforeSec (*float*, *необязательный*) – Время ожидания перед началом поиска и перефокусировки. Данный аргумент может быть полезен для отображения полезной информации перед инициализацией режима поиска

Результат:

list, список атрибутов на каждом уровне UIO объекта

pyOpenRPA.Robot.UIWeb.PageClose(*inIndexInt=None*) [исходный код]

L+,W+: Закрыть текущую вкладку или вкладку, расположенную по индексу inIndexInt. После выключения вкладки вернуть на текущую активную вкладку.

```
# UIWeb: Взаимодействие с ui web
from pyOpenRPA.Robot import UIWeb
UIWeb.PageClose(inTabIndexInt=1)
```

Параметры:

inIndexInt – Индекс вкладки, которую требуется закрыть

Type:

int

pyOpenRPA.Robot.UIWeb.PageCount() [исходный код]

L+,W+: Вернуть количество открытых вкладок в браузере

```
# UIWeb: Взаимодействие с ui web
from pyOpenRPA.Robot import UIWeb
tab_count_int = UIWeb.PageCount()
```

Результат:

Количество открытых вкладок в браузере

Тип результата:
int

pyOpenRPA.Robot.UIWeb.PageJSExecute(*inJSStr*, **inArgList*) [исходный код]

L+,W+: Отправить на выполнение на сторону браузера код JavaScript.

!ВНИМАНИЕ! Для работы необходимо проинициализировать webdriver

!ВНИМАНИЕ! Данная функция поддерживает передачу переменных в область кода JavaScript (*inArgList). Обратиться к переданным переменным из JavaScript можно с помощью ключевого слова: arguments[i], где i - это порядковый номер переданной переменной

```
# UIWeb: Взаимодействие с ui web
from pyOpenRPA.Robot import UIWeb
UIWeb.BrowserChromeStart()
UIWeb.PageOpen("https://mail.ru")
UIWeb.PageJSExecute(alert('arguments[0]);", "hello world!")
UIWeb.BrowserClose()
```

Параметры:

- **inJSStr** (*str*) – Код JavaScript, отправляемый на сторону браузера
- ***inArgList** –
  Перечисление аргументов, отправляемых на сторону браузера

Результат:
Результат отработки кода JavaScript, если он заканчивался оператором «return»

Тип результата:
str | int | bool | float

pyOpenRPA.Robot.UIWeb.PageNew(*inURLStr: str = ''*) [исходный код]

L+,W+: Открыть новую вкладку в браузере

```
# UIWeb: Взаимодействие с ui web
from pyOpenRPA.Robot import UIWeb
UIWeb.PageNew("https://pyopenrpa.ru")
```

Параметры:
inURLStr (*str, опциональный*) – URL адрес страницы, по умолчанию страница пустая, «»

pyOpenRPA.Robot.UIWeb.PageOpen(*inURLStr: str*) [исходный код]

L+,W+: Открыть страницу inURLStr в браузере и дождаться ее загрузки.

!ВНИМАНИЕ! Для работы необходимо проинициализировать webdriver

```
# UIWeb: Взаимодействие с ui web
from pyOpenRPA.Robot import UIWeb
UIWeb.BrowserChromeStart()
UIWeb.PageOpen("https://mail.ru")
UIWeb.BrowserClose()
```

Параметры:
inURLStr (*str*) – URL адрес страницы

pyOpenRPA.Robot.UIWeb.PagePrint() [исходный код]

L+,W+: Открыть окно печати браузера.

!ВНИМАНИЕ! Для работы необходимо проинициализировать webdriver

```
# UIWeb: Взаимодействие с ui web
from pyOpenRPA.Robot import UIWeb
import time
UIWeb.BrowserChromeStart()
UIWeb.PageOpen("https://mail.ru")
time.sleep(1)
UIWeb.PagePrint()
UIWeb.BrowserClose()
```

pyOpenRPA.Robot.UIWeb.PageScrollTo(*inVerticalPxInt=0, inHorizontalPxInt=0*) [исходный код]

L+,W+: Выполнить прокрутку страницы (по вертикали или по горизонтали)

!ВНИМАНИЕ! Для работы необходимо проинициализировать webdriver

```
# UIWeb: Взаимодействие с ui web
from pyOpenRPA.Robot import UIWeb
UIWeb.BrowserChromeStart()
UIWeb.PageOpen("https://mail.ru")
UIWeb.PageScrollTo(inVerticalPxInt=100)
UIWeb.BrowserClose()
```

Параметры:

- **inVerticalPxInt** (*int*, *опционально*) – Величина вертикальной прокрутки страницы в пикселях, по умолчанию 0
- **inHorizontalPxInt** (*int*, *опционально*) – Величина горизонтальной прокрутки страницы в пикселях, по умолчанию 0

pyOpenRPA.Robot.UIWeb.PageSwitch(*inTabIndexInt: int*) [исходный код]

L+,W+: Переключить вкладку по индексу inTabIndexInt

```
# UIWeb: Взаимодействие с ui web
from pyOpenRPA.Robot import UIWeb
UIWeb.PageSwitch(inTabIndexInt=1)
```

Параметры:

inTabIndexInt (*int*) – Индекс вкладки, на которую требуется переключиться

pyOpenRPA.Robot.UIWeb.SelectorConvert(*inSelector, inToTypeStr*) [исходный код]

L+,W+: Перевести селектор в заданный тип. Доступно: UIO, CSS и XPATH

```
# UIWeb: Взаимодействие с UI объектами браузера
from pyOpenRPA.Robot import UIWeb
lToType = "XPATH"
lUIOSelector = [{}]
lXPATHSelector = UIDesktop.Selector_Convert_Selector(inSelector=lUIOSelector, inToTypeStr=lToType)
```

Параметры:

- **inSelector** (*str или list*, *обязательный*) – Селектор, который необходимо преобразовать
- **inToTypeStr** (*str*, *обязательный*) – Тип, в который необходимо преобразовать селектор

Результат:

Селектор

pyOpenRPA.Robot.UIWeb.UIOAttributeDictGet(*inUIO*) [исходный код]

L+,W+: Получить словарь атрибутов UI объекта. title - наименование тэга, class_list, style_dict

```
# UIWeb: Взаимодействие с ui web
from pyOpenRPA.Robot import UIWeb
UIWeb.BrowserChromeStart()
UIWeb.PageOpen("https://mail.ru")
lUIOSelectorStr = "//*[@id='grid']/div[2]/div[2]/div[3]/div[1]/ul/li[5]/div/a"
lUIO = UIWeb.UIOSelectorFirst(inUIOSelectorStr = lUIOSelectorStr)
UIWeb.UIOAttributeDictGet(lUIO)
```

Параметры:

inUIO (*WebElement*) – UIO элемент. Получить его можно с помощью функций UIOSelectorList или UIOSelectorFirst

Результат:

Пример: {„class_list“: [], „lang“: „ru“, „style_dict“: {„–ph-color-background-accent“: None,“width“: «100%»,“color“: «black»}, „title“: „html“, «nodeName»: null}

Тип результата:

dict

pyOpenRPA.Robot.UIWeb.UIOAttributeGet(*inUIO*, *inAttributeStr*)→ str [исходный код]

L+,W+: Получить обычный (нестилевой) атрибут у UI элемента.

!ВНИМАНИЕ! Для работы необходимо проинициализировать webdriver

```
# UIWeb: Взаимодействие с ui web
from pyOpenRPA.Robot import UIWeb
UIWeb.BrowserChromeStart()
UIWeb.PageOpen("https://mail.ru")
lUIOSelectorStr = "//*[@id='grid']/div[2]/div[2]/div[3]/div[1]/ul/li[5]/div/a"
lUIO = UIWeb.UIOSelectorList(inUIOSelectorStr = lUIOSelectorStr)[0]
UIWeb.UIOAttributeGet(inUIO=lUIO, inAttributeStr = "href")
UIWeb.BrowserClose()
```

Параметры:

- **inUIO** (*WebElement*) – UIO элемент. Получить его можно с помощью функций UIOSelectorList или UIOSelectorFirst
- **inAttributeStr** (*str*) – Наименование обычного (нестилевого) атрибута

Результат:

Значение обычного (нестилевого) атрибута

Тип результата:

str

pyOpenRPA.Robot.UIWeb.UIOAttributeRemove(*inUIO*, *inAttributeStr*) [исходный код]

L+,W+: Удалить обычный (нестилевой) атрибут у UI элемента.

!ВНИМАНИЕ! Для работы необходимо проинициализировать webdriver

```
# UIWeb: Взаимодействие с ui web
from pyOpenRPA.Robot import UIWeb
UIWeb.BrowserChromeStart()
UIWeb.PageOpen("https://mail.ru")
lUIOSelectorStr = "//*[@id='grid']/div[2]/div[2]/div[3]/div[1]/ul/li[5]/div/a"
lUIO = UIWeb.UIOSelectorList(inUIOSelectorStr = lUIOSelectorStr)[0]
UIWeb.UIOAttributeRemove(lUIO, "href")
UIWeb.BrowserClose()
```

Параметры:

- **inUIO** (*WebElement*) – UIO элемент. Получить его можно с помощью функций UIOSelectorList или UIOSelectorFirst
- **inAttributeStr** (*str*) – Наименование обычного (нестилевого) атрибута

pyOpenRPA.Robot.UIWeb.UIOAttributeSet(*inUIO*, *inAttributeStr*, *inValue*) [исходный код]

L+,W+: Установить обычный (нестилевой) атрибут у UI элемента.

!ВНИМАНИЕ! Для работы необходимо проинициализировать webdriver

```
# UIWeb: Взаимодействие с ui web
from pyOpenRPA.Robot import UIWeb
UIWeb.BrowserChromeStart()
UIWeb.PageOpen("https://mail.ru")
lUIOSelectorStr = "//*[@id='grid']/div[2]/div[2]/div[3]/div[1]/ul/li[5]/div/a"
lUIO = UIWeb.UIOSelectorList(inUIOSelectorStr = lUIOSelectorStr)[0]
UIWeb.UIOAttributeSet(inUIO=lUIO, inAttributeStr = "href", inValue = "https://mail.ru")
UIWeb.BrowserClose()
```

Параметры:

- **inUIO** (*WebElement*) – UIO элемент. Получить его можно с помощью функций UIOSelectorList или UIOSelectorFirst
- **inAttributeStr** (*str*) – Наименование обычного (нестилевого) атрибута
- **inValue** (*str*) – Устанавливаемое значение обычного (нестилевого) атрибута

pyOpenRPA.Robot.UIWeb.UIOAttributeStyleGet(*inUIO*, *inAttributeStr*)→ str [исходный код]

L+,W+: Получить стилевой атрибут у UI элемента.

!ВНИМАНИЕ! Для работы необходимо проинициализировать webdriver

```
# UIWeb: Взаимодействие с ui web
from pyOpenRPA.Robot import UIWeb
UIWeb.BrowserChromeStart()
UIWeb.PageOpen("https://mail.ru")
lUIOSelectorStr = "//*[@id='grid']/div[2]/div[2]/div[3]/div[1]/ul/li[5]/div/a"
lUIO = UIWeb.UIOSelectorList(inUIOSelectorStr = lUIOSelectorStr)[0]
UIWeb.UIOAttributeStyleGet(inUIO=lUIO, inAttributeStr = "href")
UIWeb.BrowserClose()
```

Параметры:

- **inUIO** (*WebElement*) – UIO элемент. Получить его можно с помощью функций UIOSelectorList или UIOSelectorFirst
- **inAttributeStr** (*str*) – Наименование стилевого атрибута

Результат:

Значение стилевого атрибута

Тип результата:

str

pyOpenRPA.Robot.UIWeb.UIOAttributeStyleRemove(*inUIO*, *inAttributeStr: str*) [исходный код]

L+,W+: Удалить стилевой атрибут у UI элемента.

!ВНИМАНИЕ! Для работы необходимо проинициализировать webdriver

```
# UIWeb: Взаимодействие с ui web
from pyOpenRPA.Robot import UIWeb
UIWeb.BrowserChromeStart()
UIWeb.PageOpen("https://mail.ru")
lUIOSelectorStr = "//*[@id='grid']/div[2]/div[2]/div[3]/div[1]/ul/li[5]/div/a"
lUIO = UIWeb.UIOSelectorList(inUIOSelectorStr = lUIOSelectorStr)[0]
UIWeb.UIOAttributeStyleRemove(lUIO, "color")
UIWeb.BrowserClose()
```

Параметры:

- **inUIO** (*WebElement*) – UIO элемент. Получить его можно с помощью функций UIOSelectorList или UIOSelectorFirst
- **inAttributeStr** (*str*) – Наименование стилевого атрибута

pyOpenRPA.Robot.UIWeb.UIOAttributeStyleSet(*inUIO*, *inAttributeStr*, *inValue*) [исходный код]

L+,W+: Установить стилевой атрибут у UI элемента.

!ВНИМАНИЕ! Для работы необходимо проинициализировать webdriver

```
# UIWeb: Взаимодействие с ui web
from pyOpenRPA.Robot import UIWeb
UIWeb.BrowserChromeStart()
UIWeb.PageOpen("https://mail.ru")
lUIOSelectorStr = "//*[@id='grid']/div[2]/div[2]/div[3]/div[1]/ul/li[5]/div/a"
lUIO = UIWeb.UIOSelectorList(inUIOSelectorStr = lUIOSelectorStr)[0]
UIWeb.UIOAttributeStyleSet(inUIO=lUIO, inAttributeStr = "color", inValue = "grey")
UIWeb.BrowserClose()
```

Параметры:

- **inUIO** (*WebElement*) – UIO элемент. Получить его можно с помощью функций UIOSelectorList или UIOSelectorFirst
- **inAttributeStr** (*str*) – Наименование стилевого атрибута
- **inValue** (*str*) – Устанавливаемое значение стилевого атрибута

pyOpenRPA.Robot.UIWeb.UIOChildListAttributeDictGet(*inUIO*) [исходный код]

L+,W+: Получить список словарь атрибутов для детей UI объекта inUIO. title - наименование тэга, class_list, style_dict

```
# UIWeb: Взаимодействие с ui web
from pyOpenRPA.Robot import UIWeb
UIWeb.BrowserChromeStart()
UIWeb.PageOpen("https://mail.ru")
lUIOSelectorStr = "//*[@id='grid']/div[2]/div[2]/div[3]/div[1]/ul/li[5]/div/a"
lUIO = UIWeb.UIOSelectorFirst(inUIOSelectorStr = lUIOSelectorStr)
UIWeb.UIOAttributeDictGet(lUIO)
```

Параметры:

inUIO (*WebElement*) – UIO элемент. Получить его можно с помощью функций UIOSelectorList или UIOSelectorFirst

Результат:

Пример: {«ctrl_index»: 0, „class_list“: [], „lang“: „ru“, „style_dict“: {„–ph-color-background-accent“: None,“width“: «100%»,“color“: «black»}, „title“: „html“}

Тип результата:

dict

pyOpenRPA.Robot.UIWeb.UIOClick(*inUIO*) [исходный код]

L+,W+: Выполнить нажатие по элементу inUIO.

!ВНИМАНИЕ! Для работы необходимо проинициализировать webdriver

```
# UIWeb: Взаимодействие с ui web
from pyOpenRPA.Robot import UIWeb
UIWeb.BrowserChromeStart()
UIWeb.PageOpen("https://mail.ru")
lUIOSelectorStr = "//*[@id='grid']/div[2]/div[2]/div[3]/div[1]/ul/li[5]/div/a"
lUIO = UIWeb.UIOSelectorList(inUIOSelectorStr = lUIOSelectorStr)[0]
UIOClick(inUIO = lUIO)
UIWeb.BrowserClose()
```

Параметры:

inUIO (*WebElement*) – UIO элемент. Получить его можно с помощью функций UIOSelectorList или UIOSelectorFirst

pyOpenRPA.Robot.UIWeb.UIOHighlight(*inUIO, inIsFirst: bool = False, inDurationSecFloat: float = 3.0, inColorStr: str = 'green'*) [исходный код]

L+,W+: Выполнить подсвечивание UI элемента с UIO.

!ВНИМАНИЕ! Для работы необходимо проинициализировать webdriver

```
# UIWeb: Взаимодействие с ui web
from pyOpenRPA.Robot import UIWeb
UIWeb.BrowserChromeStart()
UIWeb.PageOpen("https://mail.ru")
lUIOSelectorStr = "//*[@id='grid']/div[2]/div[2]/div[3]/div[1]/ul/li[5]/div/a"
lUIO = UIOSelectorFirst(lUIOSelectorStr)
UIWeb.UIOHighlight(inUIO = lUIO)
UIWeb.BrowserClose()
```

Параметры:

- **inUIOSelectorStr** (*str*) – XPATH или CSS селектор UI элемента на web странице. Подсказки по CSS: https://devhints.io/css Подсказки по XPath: https://devhints.io/xpath
- **inIsFirst** (*bool*, *опционально*) – True - подсветить только первый элемент, который удовлетворяет селектору. По умолчанию False
- **inDurationSecFloat** (*float*, *опционально*) – Длительность подсвечивания. По умолчанию 3.0 сек.
- **inColorStr** (*str*, *опционально*) – Цвет подсвечания Варианты: «red», «blue», «grey», «yellow». По умолчанию «green» (зеленый)

pyOpenRPA.Robot.UIWeb.UIOMouseSearchInit() [исходный код]

L+,W+: Инициализирует процесс поиска UI элемента с помощью мыши. После инициализации необходимо переместить указатель мыши на искомый элемент. Для прекращения поиска необходимо использовать функцию: UIOMouseSearchReturn

!ВНИМАНИЕ! Для работы необходимо проинициализировать webdriver

```
# UIWeb: Взаимодействие с ui web
from pyOpenRPA.Robot import UIWeb
import time
UIWeb.BrowserChromeStart()
UIWeb.PageOpen("https://mail.ru")
UIWeb.UIOMouseSearchInit()
time.sleep(3)
UIWeb.UIOMouseSearchReturn()
UIWeb.BrowserClose()
```

pyOpenRPA.Robot.UIWeb.UIOMouseSearchReturn(*inStopSearchBool=True*) [исходный код]

L+,W+: Возвращает UIO объект, над которым находится указатель мыши. Предварительно должна быть вызвана функция UIWeb.UIOMouseSearchInit

!ВНИМАНИЕ! Для работы необходимо проинициализировать webdriver

```
# UIWeb: Взаимодействие с ui web
from pyOpenRPA.Robot import UIWeb
import time
UIWeb.BrowserChromeStart()
UIWeb.PageOpen("https://mail.ru")
UIWeb.UIOMouseSearchInit()
time.sleep(3)
UIWeb.UIOMouseSearchReturn()
UIWeb.BrowserClose()
```

Параметры:

inStopSearchBool – True - остановить режим поиска

Результат:

UIO объект

Тип результата:

webelement

pyOpenRPA.Robot.UIWeb.UIOSelectorActivityArgGet(*inUIOSelector*, *inActionStr*) [исходный код]

L+,W+: Сформировать преднастроенный список/словарь аргументов, передаваемый в функцию inActionStr, которая будет вызываться у объекта UIO, полученного по inUIOSelector

!ВНИМАНИЕ! ДАННАЯ ФУНКЦИОНАЛЬНОСТЬ В АВТОМАТИЧЕСКОМ РЕЖИМЕ ПОДДЕРЖИВАЕТ ВСЕ РАЗРЯДНОСТИ ПРИЛОЖЕНИЙ (32|64), КОТОРЫЕ ЗАПУЩЕНЫ В СЕСИИ. PYTHON x64 ИМЕЕТ ВОЗМОЖНОСТЬ ВЗЗАИМОДЕЙСТВИЯ С x32 UIO ОБЪЕКТАМИ, НО МЫ РЕКОМЕНДУЕМ ДОПОЛНИТЕЛЬНО ИСПОЛЬЗОВАТЬ ИНТЕРПРЕТАТОР PYTHON x32 (ПОДРОБНЕЕ СМ. ФУНКЦИЮ Configure())

```
# UIWeb: Взаимодействие с ui web
from pyOpenRPA.Robot import UIWeb
UIWeb.BrowserChromeStart()
UIWeb.PageOpen("https://mail.ru")
lUIOSelectorStr = "//*[@id='grid']/div[2]/div[2]/div[3]/div[1]/ul/li[5]/div/a"
lUIOArgDict = UIWeb.UIOSelectorActivityArgGet(inUIOSelector = lUIOSelectorStr, inActionStr="type_keys"

#Возвращаемый результат
{'Selector': [{'title': 'UIDesktop.py - ORPA - Visual Studio Code',
'class_name': 'Chrome_WidgetWin_1',
'backend': 'uia'}],
'ArgList': ['keys', None, False, False, False, True, True, True],
'ArgDict': {'keys': None,
'pause': None,
'with_spaces': False,
'with_tabs': False,
'with_newlines': False,
'turn_off_numlock': True,
'set_foreground': True,
'vk_packet': True}}
```

Параметры:

- **inUIOSelector** (*list*, *обязательный*) – UIO селектор, который определяет UIO объект, для которого будет представлен перечень доступных активностей.
- **inActionStr** (*str*, *обязательный*) – Наименование метода, к которому выполнить подбор аргументов

pyOpenRPA.Robot.UIWeb.UIOSelectorActivityListGet(*inUIOSelector*) [исходный код]

L+,W+: Получить список доступных действий/функций по UIO селектору inUIOSelector. Описание возможных активностей см. ниже.

!ВНИМАНИЕ! ДАННАЯ ФУНКЦИОНАЛЬНОСТЬ В АВТОМАТИЧЕСКОМ РЕЖИМЕ

ПОДДЕРЖИВАЕТ ВСЕ РАЗРЯДНОСТИ ПРИЛОЖЕНИЙ (32|64), КОТОРЫЕ ЗАПУЩЕНЫ В СЕСИИ. PYTHON x64 ИМЕЕТ ВОЗМОЖНОСТЬ ВЗЗАИМОДЕЙСТВИЯ С x32 UIO ОБЪЕКТАМИ, НО МЫ РЕКОМЕНДУЕМ ДОПОЛНИТЕЛЬНО ИСПОЛЬЗОВАТЬ ИНТЕРПРЕТАТОР PYTHON x32 (ПОДРОБНЕЕ СМ. ФУНКЦИЮ Configure())

```
# UIWeb: Взаимодействие с ui web
from pyOpenRPA.Robot import UIWeb
UIWeb.BrowserChromeStart()
UIWeb.PageOpen("https://mail.ru")
lUIOSelectorStr = "//*[@id='grid']/div[2]/div[2]/div[3]/div[1]/ul/li[5]/div/a"
lActivityList = UIDesktop.UIOSelectorActivityListGet(lUIOSelectorStr) # Получить список активностей по
```

Параметры:

- **inUIOSelector** (*list*, *обязательный*) – UIO селектор, который определяет UIO объект, для которого будет представлен перечень доступных активностей.
- **inFlagRaiseException** (*bool*, *опциональный*) – True - формировать ошибку exception, если платформа не обнаружина ни одного UIO объекта по заданному UIO селектору. False - обратный случай (может привести к ошибочным результатам). По умолчанию True.

pyOpenRPA.Robot.UIWeb.UIOSelectorActivityRun(*inUIOSelector*, *inActionName*, *inArgumentList=None*, *inkwArgumentObject=None*) [исходный код]

L+,W+: Выполнить активность inActionName над UIO объектом, полученным с помощью UIO селектора inUIOSelector. Описание возможных активностей см. ниже.

!ВНИМАНИЕ! ДАННАЯ ФУНКЦИОНАЛЬНОСТЬ В АВТОМАТИЧЕСКОМ РЕЖИМЕ ПОДДЕРЖИВАЕТ ВСЕ РАЗРЯДНОСТИ ПРИЛОЖЕНИЙ (32|64), КОТОРЫЕ ЗАПУЩЕНЫ В СЕСИИ. PYTHON x64 ИМЕЕТ ВОЗМОЖНОСТЬ ВЗЗАИМОДЕЙСТВИЯ С x32 UIO ОБЪЕКТАМИ, НО МЫ РЕКОМЕНДУЕМ ДОПОЛНИТЕЛЬНО ИСПОЛЬЗОВАТЬ ИНТЕРПРЕТАТОР PYTHON x32 (ПОДРОБНЕЕ СМ. ФУНКЦИЮ Configure())

```
# UIWeb: Взаимодействие с ui web
from pyOpenRPA.Robot import UIWeb
UIWeb.BrowserChromeStart()
UIWeb.PageOpen("https://mail.ru")
lUIOSelectorStr = "//*[@id='grid']/div[2]/div[2]/div[3]/div[1]/ul/li[5]/div/a"
lActivityResult = UIWeb.UIOSelectorActivityRun(lUIOSelectorStr, "click") # выполнить действие над UIO
```

Параметры:

- **inUIOSelector** (*list*, *обязательный*) – UIO селектор, который определяет UIO объект, для которого будет представлен перечень доступных активностей.
- **inActionName** (*str*, *обязательный*) – наименование активности, которую требуется выполнить над UIO объектом
- **inArgumentList** (*list*, *необязательный*) – список передаваемых неименованных аргументов в функцию inActionName
- **inkwArgumentObject** (*dict*, *необязательный*) – словарь передаваемых именованных аргументов в функцию inActionName

Результат:

возвращает результат запускаемой функции с наименованием inActionName над UIO объектом

pyOpenRPA.Robot.UIWeb.UIOSelectorChildListAttributeDictGet(*inUIOSelector=None*) [исходный код]

L+,W+: Получить список словарей атриботов детей UI объекта, определенного по inUIOSelector

```
# UIWeb: Взаимодействие с ui web
from pyOpenRPA.Robot import UIWeb
UIWeb.BrowserChromeStart()
UIWeb.PageOpen("https://mail.ru")
lUIOSelectorStr = "//*[@id='grid']/div[2]/div[2]/div[3]/div[1]/ul/li[5]/div/a"
lUIO = UIWeb.UIOSelectorFirst(inUIOSelectorStr = lUIOSelectorStr)
UIWeb.UIOAttributeDictGet(lUIO)
```

Параметры:

inUIOSelector (*str*) – XPATH или CSS селектор UI элемента на web странице. Подсказки по CSS: https://devhints.io/css Подсказки по XPath: https://devhints.io/xpath

Результат:

Вернуть список детей [{„class_list“: [], „lang“: „ru“, „style_dict“: {„–ph-color-background-accent“: None,“width“: «100%»,“color“: «black»}, „title“: „html“}]

Тип результата:

list

pyOpenRPA.Robot.UIWeb.UIOSelectorClick(*inUIOSelectorStr: str*) [исходный код]

L+,W+: Выполнить нажатие по элементу с селектором inUIOSelectorStr.

!ВНИМАНИЕ! Для работы необходимо проинициализировать webdriver

```
# UIWeb: Взаимодействие с ui web
from pyOpenRPA.Robot import UIWeb
UIWeb.BrowserChromeStart()
UIWeb.PageOpen("https://mail.ru")
lUIOSelectorStr = "//*[@id='grid']/div[2]/div[2]/div[3]/div[1]/ul/li[5]/div/a"
UIWeb.UIOSelectorClick(inUIOSelectorStr = lUIOSelectorStr)
UIWeb.BrowserClose()
```

Параметры:

inUIOSelectorStr (*str*) – XPATH или CSS селектор UI элемента на web странице. Подсказки по CSS: https://devhints.io/css Подсказки по XPath: https://devhints.io/xpath

pyOpenRPA.Robot.UIWeb.UIOSelectorDetect(*inUIOSelectorStr: str*)→ str [исходный код]

L+,W+: Идентифицировать стиль селектора (CSS или XPATH или UIO)

```
# UIWeb: Взаимодействие с ui web
from pyOpenRPA.Robot import UIWeb
lUIOSelectorStr = "#grid > div.grid-middle > div.grid__main-col.svelte-2y66pa > div.grid_newscol.grid_
lUIOSelectorStr = "//*[@id='grid']/div[2]/div[2]/div[3]/div[1]/ul/li[5]/div/a"
lResultStr = UIWeb.UIOSelectorDetect(inUIOSelectorStr = lUIOSelectorStr)
```

Параметры:

inUIOSelectorStr (*str*) – XPATH или CSS селектор UI объекта на web странице. Подсказки по CSS: https://devhints.io/css Подсказки по XPath: https://devhints.io/xpath

Результат:

«CSS» или «XPATH» или «UIO»

Тип результата:

str

pyOpenRPA.Robot.UIWeb.UIOSelectorFirst(*inUIOSelectorStr=None*, *inUIO=None*)→ list [исходный код]

L+,W+: Получить UIO объект по UIO селектору.

!ВНИМАНИЕ! Для работы необходимо проинициализировать webdriver

```
# UIWeb: Взаимодействие с ui web
from pyOpenRPA.Robot import UIWeb
UIWeb.BrowserChromeStart()
UIWeb.PageOpen("https://mail.ru")
lUIOSelectorStr = "//*[@id='grid']/div[2]/div[2]/div[3]/div[1]/ul/li[5]/div/a"
lUIO = UIWeb.UIOSelectorFirst(inUIOSelectorStr = lUIOSelectorStr)
UIWeb.BrowserClose()
```

Параметры:

- **inUIOSelectorStr** (*str*) – XPATH или CSS селектор UI объекта на web странице. Подсказки по CSS: https://devhints.io/css Подсказки по XPath: https://devhints.io/xpath
- **inUIO** (*WebElement*, *опционально*) – Объект UIO, от которого выполнить поиск UIO объектов по селектору, по умолчанию None

Результат:

Первый подходящий UIO объект

Тип результата:

UIO объект

pyOpenRPA.Robot.UIWeb.UIOSelectorFocusHighlight(*inUIOSelectorStr: str, inIsFirst: bool = False, inDurationSecFloat: float = 3.0, inColorStr: str = 'green'*) [исходный код]

L+,W+: Выполнить подсвечивание UI элемента с селектором inUIOSelectorStr.

!ВНИМАНИЕ! Для работы необходимо проинициализировать webdriver

```
# UIWeb: Взаимодействие с ui web
from pyOpenRPA.Robot import UIWeb
UIWeb.BrowserChromeStart()
UIWeb.PageOpen("https://mail.ru")
lUIOSelectorStr = "//*[@id='grid']/div[2]/div[2]/div[3]/div[1]/ul/li[5]/div/a"
UIWeb.UIOSelectorFocusHighlight(inUIOSelectorStr = lUIOSelectorStr)
UIWeb.BrowserClose()
```

Параметры:

- **inUIOSelectorStr** (*str*) – XPATH или CSS селектор UI элемента на web странице. Подсказки по CSS: https://devhints.io/css Подсказки по XPath: https://devhints.io/xpath
- **inIsFirst** (*bool*, *опционально*) – True - подсветить только первый элемент, который удовлетворяет селектору. По умолчанию False
- **inDurationSecFloat** (*float*, *опционально*) – Длительность подсвечивания. По умолчанию 3.0 сек.
- **inColorStr** (*str*, *опционально*) – Цвет подсвечания Варианты: «red», «blue», «grey», «yellow». По умолчанию «green» (зеленый)

pyOpenRPA.Robot.UIWeb.UIOSelectorHighlight(*inUIOSelectorStr: str, inIsFirst: bool = False, inDurationSecFloat: float = 3.0, inColorStr: str = 'green'*) [исходный код]

L+,W+: Выполнить подсвечивание UI элемента с селектором inUIOSelectorStr.

!ВНИМАНИЕ! Для работы необходимо проинициализировать webdriver

```
# UIWeb: Взаимодействие с ui web
from pyOpenRPA.Robot import UIWeb
UIWeb.BrowserChromeStart()
UIWeb.PageOpen("https://mail.ru")
lUIOSelectorStr = "//*[@id='grid']/div[2]/div[2]/div[3]/div[1]/ul/li[5]/div/a"
UIWeb.UIOSelectorHighlight(inUIOSelectorStr = lUIOSelectorStr)
UIWeb.BrowserClose()
```

Параметры:

- **inUIOSelectorStr** (*str*) – XPATH или CSS селектор UI элемента на web странице. Подсказки по CSS: https://devhints.io/css Подсказки по XPath: https://devhints.io/xpath
- **inIsFirst** (*bool*, *опционально*) – True - подсветить только первый элемент, который удовлетворяет селектору. По умолчанию False
- **inDurationSecFloat** (*float*, *опционально*) – Длительность подсвечивания. По умолчанию 3.0 сек.
- **inColorStr** (*str*, *опционально*) – Цвет подсвечания Варианты: «red», «blue», «grey», «yellow». По умолчанию «green» (зеленый)

pyOpenRPA.Robot.UIWeb.UIOSelectorLevelInfoList(*inUIOSelector*) [исходный код]

L+,W+: Получить список свойств всех уровней до UI объекта, обнаруженного с помощью селектора inUIOSelector.

!ВНИМАНИЕ! ДАННАЯ ФУНКЦИОНАЛЬНОСТЬ В АВТОМАТИЧЕСКОМ РЕЖИМЕ ПОДДЕРЖИВАЕТ ВСЕ РАЗРЯДНОСТИ ПРИЛОЖЕНИЙ (32|64), КОТОРЫЕ ЗАПУЩЕНЫ В СЕСИИ. PYTHON x64 ИМЕЕТ ВОЗМОЖНОСТЬ ВЗЗАИМОДЕЙСТВИЯ С x32 UIO ОБЪЕКТАМИ, НО МЫ РЕКОМЕНДУЕМ ДОПОЛНИТЕЛЬНО ИСПОЛЬЗОВАТЬ ИНТЕРПРЕТАТОР PYTHON x32 (ПОДРОБНЕЕ СМ. ФУНКЦИЮ Configure())

```
# UIWeb: Взаимодействие с ui web
from pyOpenRPA.Robot import UIWeb
UIWeb.BrowserChromeStart()
UIWeb.PageOpen("https://mail.ru")
lUIOSelectorStr = "//*[@id='grid']/div[2]/div[2]/div[3]/div[1]/ul/li[5]/div/a"
lUIOLevelList = UIWeb.UIOSelectorLevelInfoList(inUIOSelector = inUIOSelector)
```

Параметры:

inUIOSelector (*list*, *обязательный*) – Селектор на UIO объект (CSS | XPATH | UIO).

Результат:

список уровней UIO объектов

pyOpenRPA.Robot.UIWeb.UIOSelectorList(*inUIOSelectorStr=None*, *inUIO=None*)→ list [исходный код]

L+,W+: Получить список UIO объектов по UIO селектору.

!ВНИМАНИЕ! Для работы необходимо проинициализировать webdriver

```
# UIWeb: Взаимодействие с ui web
from pyOpenRPA.Robot import UIWeb
UIWeb.BrowserChromeStart()
UIWeb.PageOpen("https://mail.ru")
lUIOSelectorStr = "//*[@id='grid']/div[2]/div[2]/div[3]/div[1]/ul/li[5]/div/a"
lUIOList = UIWeb.UIOSelectorList(inUIOSelectorStr = lUIOSelectorStr)
UIWeb.BrowserClose()
```

Параметры:

- **inUIOSelectorStr** (*str*) – XPATH или CSS селектор UI объекта на web странице. Подсказки по CSS: https://devhints.io/css Подсказки по XPath: https://devhints.io/xpath
- **inUIO** (*WebElement*, *опционально*) – Объект UIO, от которого выполнить поиск UIO объектов по селектору, по умолчанию None

Результат:

Список UIO объектов

Тип результата:

list

pyOpenRPA.Robot.UIWeb.UIOSelectorListAttributeDictGet(*inUIOSelector=None*) [исходный код]

L+,W+: Получить словарь атрибутов для UI объектов, которые удовлетворяют inUIOSelector селектору

```
# UIWeb: Взаимодействие с ui web
from pyOpenRPA.Robot import UIWeb
UIWeb.BrowserChromeStart()
UIWeb.PageOpen("https://mail.ru")
lUIOSelectorStr = "//*[@id='grid']/div[2]/div[2]/div[3]/div[1]/ul/li[5]/div/a"
lAttributeList = UIWeb.UIOSelectorListAttributeDictGet(inUIOSelector = lUIOSelectorStr)
```

Параметры:

inUIOSelector (*str*) – XPATH или CSS селектор UI элемента на web странице. Подсказки по CSS: https://devhints.io/css Подсказки по XPath: https://devhints.io/xpath

Результат:

Список словарей UI объектов по селектору [{„class_list“: [], „lang“: „ru“, „style_dict“: {„–ph-color-background-accent“: None,“width“: «100%»,“color“: «black»}, „title“: „html“}]

Тип результата:

list

pyOpenRPA.Robot.UIWeb.UIOSelectorSetValue(*inUIOSelectorStr: str, inValue: str*) [исходный код]

L+,W+: Установить значение элемента с селектором inUIOSelectorStr.

!ВНИМАНИЕ! Для работы необходимо проинициализировать webdriver

```
# UIWeb: Взаимодействие с ui web
from pyOpenRPA.Robot import UIWeb
UIWeb.BrowserChromeStart()
UIWeb.PageOpen("https://www.google.com/")
lUIOSelectorStr = "/html/body/div[1]/div[3]/form/div[1]/div[1]/div[1]/div/div[2]/input"
lValue = "pyOpenRPA"
UIWeb.UIOSelectorSetValue(inUIOSelectorStr = lUIOSelectorStr, inValue = lValue)
UIWeb.BrowserClose()
```

Параметры:

- **inUIOSelectorStr** (*str*) – XPATH или CSS селектор UI элемента на web странице. Подсказки по CSS: https://devhints.io/css Подсказки по XPath: https://devhints.io/xpath
- **inValue** (*str*) – Значение, которое необходимо установить

pyOpenRPA.Robot.UIWeb.UIOSelectorWaitAppear(*inUIOSelectorStr: str, inWaitSecFloat: float = 60, inWaitIntervalSecFloat: float = 1.0*) [исходный код]

L+,W+: Ожидать появление UI элемента на веб странице (блокирует выполнение потока), заданного по UIO селектору inUIOSelectorStr. Выполнять ожидание на протяжении inWaitSecFloat (по умолчанию 60 сек.). Проверка производится с интервалом inWaitIntervalSecFloat (по умолчанию 1 сек.)

!ВНИМАНИЕ! Для работы необходимо проинициализировать webdriver

```
# UIWeb: Взаимодействие с ui web
from pyOpenRPA.Robot import UIWeb
UIWeb.BrowserChromeStart()
UIWeb.PageOpen("https://mail.ru")
lUIOSelectorStr = "//*[@id='grid']/div[2]/div[2]/div[3]/div[1]/ul/li[5]/div/a"
lAppearUIOList = UIWeb.UIOSelectorWaitAppear(inUIOSelectorStr = lUIOSelectorStr)
UIWeb.BrowserClose()
```

Параметры:

- **inUIOSelectorStr** (*str*) – XPATH или CSS селектор UI элемента на web странице. Подсказки по CSS: https://devhints.io/css Подсказки по XPath: https://devhints.io/xpath
- **inWaitSecFloat** (*float*, *опциональный*) – Время ожидания на исчезновение UI элемента, по

умолчанию UIO_WAIT_SEC_FLOAT (60 сек)

- **inWaitIntervalSecFloat** (*float*, *опциональный*) – Интервал проверки исчезновения, по умолчанию UIO_WAIT_INTERVAL_SEC_FLOAT (1 сек)

Исключение:

Exception – Время ожидания превышено

Результат:

Список UI элементов, которые удовлетворяют селектору и появились на странице

Тип результата:

list

---

pyOpenRPA.Robot.UIWeb.UIOSelectorWaitDisappear(*inUIOSelectorStr: str, inWaitSecFloat: float = 60, inWaitIntervalSecFloat: float = 1.0*) [исходный код]

L+,W+: Ожидать исчезновение UI элемента с веб страницы (блокирует выполнение потока), заданного по UIO селектору inUIOSelectorStr. Выполнять ожидание на протяжении inWaitSecFloat (по умолчанию 60 сек.). Проверка производится с интервалом inWaitIntervalSecFloat (по умолчанию 1 сек.)

!ВНИМАНИЕ! Для работы необходимо проинициализировать webdriver

```
# UIWeb: Взаимодействие с ui web
from pyOpenRPA.Robot import UIWeb
UIWeb.BrowserChromeStart()
UIWeb.PageOpen("https://mail.ru")
lUIOSelectorStr = "//*[@id='grid']/div[2]/div[2]/div[3]/div[1]/ul/li[5]/div/a"
UIWeb.UIOSelectorWaitDisappear(inUIOSelectorStr = lUIOSelectorStr)
UIWeb.BrowserClose()
```

Параметры:

- **inUIOSelectorStr** (*str*) – XPATH или CSS селектор UI элемента на web странице. Подсказки по CSS: https://devhints.io/css Подсказки по XPath: https://devhints.io/xpath
- **inWaitSecFloat** (*float*, *опциональный*) – Время ожидания на исчезновение UI элемента, по умолчанию UIO_WAIT_SEC_FLOAT (60 сек)
- **inWaitIntervalSecFloat** (*float*, *опциональный*) – Интервал проверки исчезновения, по умолчанию UIO_WAIT_INTERVAL_SEC_FLOAT (1 сек)

Исключение:

Exception – Время ожидания превышено

---

pyOpenRPA.Robot.UIWeb.UIOSelector_To_XPATH(*inUIOSelector=None*) [исходный код]

L+,W+: Выполнить конвертацию селектора из UIO в XPATH

```
# UIWeb: Взаимодействие с UI объектами вкладки
from pyOpenRPA.Robot import UIWeb
lUIOSelector = [{'depth_start': 1, 'depth_end': 99, 'id': 'grid'},
    {'ctrl_index': 2},
    {'depth_start': 1, 'depth_end': 99, 'title': 'div', 'ctrl_index': 2},
    {'title': 'div', 'ctrl_index': 3},
    {'title': 'div', 'ctrl_index': 1},
    {'title': 'ul'},
    {'title': 'li', 'ctrl_index': 5},
    {'title': 'div'},
    {'title': 'a'}]
lXPATHSelector = UIWeb.UIOSelector_To_XPATH(inUIOSelector=lUIOSelector)
```

Параметры:

inUIOSelector (*list*, *обязательный*) – Селектор в формате UIO

---

pyOpenRPA.Robot.UIWeb.UIOTextGet(*inUIO*)→ str [исходный код]

L+,W+: Получить текст UI элемента.

!ВНИМАНИЕ! Для работы необходимо проинициализировать webdriver

```
# UIWeb: Взаимодействие с ui web
from pyOpenRPA.Robot import UIWeb
UIWeb.BrowserChromeStart()
UIWeb.PageOpen("https://mail.ru")
lUIOSelectorStr = "//*[@id='grid']/div[2]/div[2]/div[3]/div[1]/ul/li[5]/div/a"
lUIO = UIWeb.UIOSelectorList(inUIOSelectorStr = lUIOSelectorStr)[0]
lTextStr = UIWeb.UIOTextGet(inUIO=lUIO)
UIWeb.BrowserClose()
```

Параметры:

inUIO (*WebElement*) – UIO элемент. Получить его можно с помощью функций UIOSelectorList или UIOSelectorFirst

Результат:

Текст UI элемента

Тип результата:

str

pyOpenRPA.Robot.UIWeb.XPATH_To_UIOSelector(*inXPATHSelector=None*) [исходный код]

L+,W+: Выполнить конвертацию селектора из XPATH в UIO

```
# UIDesktop: Взаимодействие с UI объектами приложений
from pyOpenRPA.Robot import UIWeb
lXPATHSelector = "//*[@id='grid']/*[2]/div[2]/div[3]/div[1]/ul/li[5]/div/a"
lUIOSelector = UIDesktop.XPATH_To_UIOSelector(inXPATHSelector=lXPATHSelector)
```

Параметры:

inUIOSelector (*str, обязательный*) – Селектор в формате XPATH

Быстрая навигация

- Сообщество pyOpenRPA (telegram)
- Сообщество pyOpenRPA (tenchat)
- Сообщество pyOpenRPA (вконтакте)
- Презентация pyOpenRPA
- Портал pyOpenRPA
- Репозиторий pyOpenRPA

.. v1.4.0 replace:: v1.4.0

Предыдущая Следующая

4. Функции Keyboard

Общее

Клавиатура - это главный текстовый инструмент, который обладает 100% точностью передачи данных. С его помощью можно отправлять сообщения, ожидать нажатия и выполнять различные комбинации клавиш. На этой странице представлена вся необходимая информация по управлению клавиатурой со стороны программного робота RPA.

В отличие от многих RPA платформ, pyOpenRPA обладает функциями, которые не зависят от текущей раскладки клавиатуры. За счет этого надежность и стабильность программного робота существенно возрастает.

ВНИМАНИЕ! ПРИ ВЫЗОВЕ ФУНКЦИЙ ОБРАЩАЙТЕ ВНИМАНИЕ НА РЕГИСТР.

Доп. настройки в LINUX

Используется компонент setxkbmap: apt-get install x11-xkb-utils (компонент для взаимодействия с клавиатурой, https://command-not-found.com/setxkbmap)

Особенности ОС Linux позволяют выполнять ввод разноязычного текста только через переключение раскладок клавиатуры в режиме реального времени По умолчанию установлены следующие раскладки для Русского и английского языков: Keyboard.KEY_RUS_LAYOUT = «ru» # NEED FOR LINUX (FOR LAYOUT SWITCH) Keyboard.KEY_ENG_LAYOUT = «us» # NEED FOR LINUX (FOR LAYOUT SWITCH)

Проверить, что данные раскладки работают корректно, можно с помощью следующей команды в терминале: setxkbmap -layout us,ru -option grp:alt_shift_toggle После ввода попробуйте ввести английские символы, после чего переключиться на другой язык с помощью комбинации клавиш Alt+Shift

Если у вас используются другие layout, то вы можете указать их в переменных Keyboard.KEY_RUS_LAYOUT и Keyboard.KEY_ENG_LAYOUT для русского и английского языков соответственно.

Примеры использования

```
# Keyboard: Взаимодействие с клавиатурой
from pyOpenRPA.Robot import Keyboard
Keyboard.HotkeyCombination(Keyboard.KEY_HOT_WIN_LEFT,Keyboard.KEY_ENG_R)
Keyboard.Write("cmd")
Keyboard.Send(Keyboard.KEY_HOT_ENTER, inWaitAfterSecFloat=0.6)
Keyboard.Write("echo %time%")
Keyboard.Send(Keyboard.KEY_HOT_ENTER)
Keyboard.HotkeyCombination(Keyboard.KEY_HOT_CTRL_LEFT, Keyboard.KEY_ENG_A, inWaitAfterSecFloat=0.6)
Keyboard.HotkeyCombination(Keyboard.KEY_HOT_CTRL_LEFT, Keyboard.KEY_ENG_C, inWaitAfterSecFloat=0.6)
```

Коды клавиш см. ниже

Описание функций

Описание каждой функции начинается с обозначения L+,W+, что означает, что функция поддерживается в ОС Linux (L) и Windows (W)

Functions:

`Down` (inKeyInt[, inWaitAfterSecFloat])	L+,W+: Нажать (опустить) клавишу.
`HotkeyCombination` (*inKeyList[, ...])	L+,W+: Получает перечень клавиш для одновременного нажатия.
`HotkeyCtrlA_CtrlC` ([inWaitAfterSecFloat])	L+,W+: Выполнить выделение текста, после чего скопировать его в
`HotkeyCtrlV` ([inWaitAfterSecFloat])	L+,W+: Выполнить вставку текста из буфера обмена ВНИМАНИЕ! Н
`IsDown` (inKeyInt)	L+,W+: Проверить, опущена ли клавиша.
`Send` (inKeyInt[, inDoPressBool, ...])	L+,W+: Имитация нажатия/отпускания любой физической клавиши.
`Up` (inKeyInt[, inWaitAfterSecFloat])	L+,W+: Отпустить (поднять) клавишу.
`Wait` (inKeyInt[, inWaitAfterSecFloat])	L-,W+: Блокирует осуществление программы, пока данная обозначе
`Write` (inTextStr[, inDelayFloat, ...])	L+,W+: Печатает текст, который был передан в переменной inTextSt

Data:

`KEY_ENG_COLON`	
`KEY_HOT_COLON`	
`KEY_RUS_Ж`	

pyOpenRPA.Robot.Keyboard.Down(*inKeyInt: int, inWaitAfterSecFloat: float = 0.4*)→ None [исходный код]

L+,W+: Нажать (опустить) клавишу. Если клавиша уже была опущена, то ничего не произойдет.

ВНИМАНИЕ! ПРИ ПОПЫТКЕ ПЕЧАТИ ТЕКСТА БУДЕТ УЧИТЫВАТЬ ТЕКУЩУЮ РАСКЛАДКУ КЛАВИАТУРЫ. ДЛЯ ПЕЧАТИ ТЕКСТА ИСПОЛЬЗУЙ Write!

ВНИМАНИЕ! ФУНКЦИЯ МОЖЕТ ОТРАБОТАТЬ НЕКОРРЕКТНО В ТОМ СЛУЧАЕ, ЕСЛИ ДЕЙСТВИЕ ПРОИСХОДИТ СРАЗУ ПОСЛЕ НАЖАТИЯ КЛАВИШИ ENTER (ИСПОЛЬЗУЙТЕ SLEEP)

```
# Keyboard: Взаимодействие с клавиатурой
from pyOpenRPA.Robot import Keyboard
Keyboard.Down(Keyboard.KEY_ENG_A) # Отпустить клавишу A.
```

Параметры:

- **inKeyInt** (*int*) – Перечень клавиш см. в разделе «Коды клавиш». Пример: KEY_HOT_CTRL_LEFT, KEY_ENG_A
- **inWaitAfterSecFloat** (*float*, *опциональный*) – Количество секунд, которые ожидать после выполнения операции. По умолчанию установлено в настройках модуля Keyboard (базовое значение 0.4)

pyOpenRPA.Robot.Keyboard.HotkeyCombination(**inKeyList, inDelaySecFloat=0.3, inWaitAfterSecFloat: float = 0.4*)
[исходный код]

L+,W+: Получает перечень клавиш для одновременного нажатия. Между нажатиями программа ожидания время inDelaySecFloat ВНИМАНИЕ! НЕ ЗАВИСИТ ОТ ТЕКУЩЕЙ РАСКЛАДКИ

КЛАВИАТУРЫ

```
# Keyboard: Взаимодействие с клавиатурой
from pyOpenRPA.Robot import Keyboard
Keyboard.HotkeyCombination(Keyboard.KEY_HOT_CTRL_LEFT,Keyboard.KEY_ENG_A) # Ctrl + a
Keyboard.HotkeyCombination(Keyboard.KEY_HOT_CTRL_LEFT,Keyboard.KEY_ENG_C) # Ctrl + c
Keyboard.HotkeyCombination(Keyboard.KEY_HOT_CTRL_LEFT,Keyboard.KEY_ENG_A)
Keyboard.HotkeyCombination(Keyboard.KEY_HOT_ALT_LEFT,Keyboard.KEY_HOT_TAB, Keyboard.KEY_HOT_TAB)
```

Параметры:

- **inKeyList** – Список клавиш для одновременного нажатия. Перечень клавиш см. в разделе «Коды клавиш». Пример: KEY_HOT_CTRL_LEFT,KEY_ENG_A
- **inDelaySecFloat** (*float*, *опциональный*) – Интервал между нажатиями. Необходим в связи с тем, что операционной системе требуется время на реакцию на нажатие клавиш, по умолчанию: 0.3
- **inWaitAfterSecFloat** (*float*, *опциональный*) – Количество секунд, которые ожидать после выполнения операции. По умолчанию установлено в настройках модуля Keyboard (базовое значение 0.4)

pyOpenRPA.Robot.Keyboard.HotkeyCtrlA_CtrlC(*inWaitAfterSecFloat: float = 0.4*)→ None [исходный код]

L+,W+: Выполнить выделение текста, после чего скопировать его в буфер обмена ВНИМАНИЕ! НЕ ЗАВИСИТ ОТ ТЕКУЩЕЙ РАСКЛАДКИ КЛАВИАТУРЫ

```
# Keyboard: Взаимодействие с клавиатурой
from pyOpenRPA.Robot import Keyboard
Keyboard.HotkeyCtrlA_CtrlC() # Отправить команды: выделить все, скопировать в буфер обмена
```

Параметры:

inWaitAfterSecFloat (*float*, *опциональный*) – Количество секунд, которые ожидать после выполнения операции. По умолчанию установлено в настройках модуля Keyboard (базовое значение 0.4)

pyOpenRPA.Robot.Keyboard.HotkeyCtrlV(*inWaitAfterSecFloat: float = 0.4*)→ None [исходный код]

L+,W+: Выполнить вставку текста из буфера обмена ВНИМАНИЕ! НЕ ЗАВИСИТ ОТ ТЕКУЩЕЙ РАСКЛАДКИ КЛАВИАТУРЫ

```
# Keyboard: Взаимодействие с клавиатурой
from pyOpenRPA.Robot import Keyboard
Keyboard.HotkeyCtrlV()
```

Параметры:

inWaitAfterSecFloat (*float*, *опциональный*) – Количество секунд, которые ожидать после выполнения операции. По умолчанию установлено в настройках модуля Keyboard (базовое значение 0.4)

pyOpenRPA.Robot.Keyboard.IsDown(*inKeyInt: int*)→ bool [исходный код]

L+,W+: Проверить, опущена ли клавиша. Вернет True если опущена; False если поднята.

ВНИМАНИЕ! ПРИ ПОПЫТКЕ ПЕЧАТИ ТЕКСТА БУДЕТ УЧИТЫВАТЬ ТЕКУЩУЮ РАСКЛАДКУ КЛАВИАТУРЫ. ДЛЯ ПЕЧАТИ ТЕКСТА ИСПОЛЬЗУЙ Write!

```
# Keyboard: Взаимодействие с клавиатурой
from pyOpenRPA.Robot import Keyboard
lKeyAIsPressedBool = Keyboard.IsDown(Keyboard.KEY_ENG_A) # Проверить, опущена ли клавиша A.
```

Параметры:

inKeyInt (*int*) – Перечень клавиш см. в разделе «Коды клавиш». Пример: KEY_HOT_CTRL_LEFT, KEY_ENG_A

pyOpenRPA.Robot.Keyboard.KEY_ENG_COLON= *39*

pyOpenRPA.Robot.Keyboard.KEY_HOT_COLON= *39*

pyOpenRPA.Robot.Keyboard.KEY_RUS_Ж= *39*

pyOpenRPA.Robot.Keyboard.Send(*inKeyInt: int, inDoPressBool: bool = True, inDoReleaseBool: bool = True, inWaitAfterSecFloat: float = 0.4*)→ None [исходный код]

L+,W+: Имитация нажатия/отпускания любой физической клавиши. Посылает событие в операционную систему, которые выполняет нажатие и отпускание данной клавиши

ВНИМАНИЕ! ПРИ ПОПЫТКЕ ПЕЧАТИ ТЕКСТА БУДЕТ УЧИТЫВАТЬ ТЕКУЩУЮ РАСКЛАДКУ КЛАВИАТУРЫ. ДЛЯ ПЕЧАТИ ТЕКСТА ИСПОЛЬЗУЙ Write!

```
# Keyboard: Взаимодействие с клавиатурой
from pyOpenRPA.Robot import Keyboard
Keyboard.Send(Keyboard.KEY_ENG_A) # Нажать клавишу A. Если будет активна русская раскладка, то будет б
```

Параметры:

- **inKeyInt** (*int*) – Перечень клавиш см. в разделе «Коды клавиш». Пример: KEY_HOT_CTRL_LEFT,KEY_ENG_A
- **inDoPressBool** (*bool*, *опциональный*) – Выполнить событие нажатия клавиши, По умолчанию True
- **inDoReleaseBool** (*bool*, *опциональный*) – Выполнить событие отпускания клавиши, По умолчанию True
- **inWaitAfterSecFloat** (*float*, *опциональный*) – Количество секунд, которые ожидать после выполнения операции. По умолчанию установлено в настройках модуля Keyboard (базовое значение 0.4)

pyOpenRPA.Robot.Keyboard.Up(*inKeyInt: int, inWaitAfterSecFloat: float = 0.4*)→ None [исходный код]

L+,W+: Отпустить (поднять) клавишу. Если клавиша уже была поднята, то ничего не произойдет.

ВНИМАНИЕ! ПРИ ПОПЫТКЕ ПЕЧАТИ ТЕКСТА БУДЕТ УЧИТЫВАТЬ ТЕКУЩУЮ РАСКЛАДКУ КЛАВИАТУРЫ. ДЛЯ ПЕЧАТИ ТЕКСТА ИСПОЛЬЗУЙ Write!

ВНИМАНИЕ! ФУНКЦИЯ МОЖЕТ ОТРАБОТАТЬ НЕКОРРЕКТНО В ТОМ СЛУЧАЕ, ЕСЛИ ДЕЙСТВИЕ ПРОИСХОДИТ СРАЗУ ПОСЛЕ НАЖАТИЯ КЛАВИШИ ENTER (ИСПОЛЬЗУЙТЕ SLEEP)

```
# Keyboard: Взаимодействие с клавиатурой
from pyOpenRPA.Robot import Keyboard
Keyboard.Up(Keyboard.KEY_ENG_A) # Отпустить клавишу A.
```

Параметры:

- **inKeyInt** (*int*) – Перечень клавиш см. в разделе «Коды клавиш». Пример: KEY_HOT_CTRL_LEFT, KEY_ENG_A
- **inWaitAfterSecFloat** (*float*, *опциональный*) – Количество секунд, которые ожидать после выполнения операции. По умолчанию установлено в настройках модуля Keyboard (базовое значение 0.4)

pyOpenRPA.Robot.Keyboard.Wait(*inKeyInt: int, inWaitAfterSecFloat: float = 0.4*) [исходный код]

L-,W+: Блокирует осуществление программы, пока данная обозначенная клавиша не будет нажата. ВНИМАНИЕ! НЕ ЗАВИСИТ ОТ ТЕКУЩЕЙ РАСКЛАДКИ КЛАВИАТУРЫ. ОЖИДАЕТ НАЖАТИЕ СООТВЕТСВУЮЩЕЙ ФИЗИЧЕСКОЙ КЛАВИШИ

```
# Keyboard: Взаимодействие с клавиатурой
from pyOpenRPA.Robot import Keyboard
Keyboard.Wait(Keyboard.KEY_ENG_A) # Ждать нажатие клавиши A.
```

Параметры:

- **inKeyInt** (*int*) – Перечень клавиш см. в разделе «Коды клавиш». Пример: KEY_HOT_CTRL_LEFT,KEY_ENG_A
- **inWaitAfterSecFloat** (*float*, *опциональный*) – Количество секунд, которые ожидать после выполнения операции. По умолчанию установлено в настройках модуля Keyboard (базовое значение 0.4)

pyOpenRPA.Robot.Keyboard.Write(*inTextStr: str, inDelayFloat: float = 0, inRestoreStateAfterBool: bool = True, inExactBool: Optional[bool] = None, inWaitAfterSecFloat: float = 0.4*) [исходный код]

L+,W+: Печатает текст, который был передан в переменной inTextStr (поддерживает передачу в одной строке символов разного языка). Не зависит от текущей раскладки клавиатуры! Посылает искусственные клавишные события в ОС, моделируя печать данного текста. Знаки, не доступные на клавиатуре, напечатаны как явный unicode знаки, использующие определенную для ОС функциональность, такие как alt+codepoint. Чтобы гарантировать текстовую целостность, все в настоящее время нажатые ключи выпущены прежде текст напечатан, и модификаторы восстановлены впоследствии.

ВНИМАНИЕ! ПЕЧАТАЕТ ЛЮБУЮ СТРОКУ, ДАЖЕ В СОЧЕТАНИИ НЕСКОЛЬКИХ ЯЗЫКОВ ОДНОВРЕМЕННО. ДЛЯ РАБОТЫ С ГОРЯЧИМИ КЛАВИШАМИ ИСПОЛЬЗУЙ ФУНКЦИЮ Send, Up, Down, HotkeyCombination

ВНИМАНИЕ! В LINUX НЕ ДЕЙСТВУЮТ СЛЕДУЮЩИЕ ПАРАМЕТРЫ: inRestoreStateAfterBool, inExactBool

```
# Keyboard: Взаимодействие с клавиатурой
from pyOpenRPA.Robot import Keyboard
Keyboard.Write("Привет мой милый мир! Hello my dear world!")
```

Параметры:

- **inTextStr** (*str*) – Текст, отправляемый на печать. Не зависит от текущей раскладки клавиатуры!
- **inDelayFloat** (*float*, *опциональный*) – Число секунд, которое ожидать между нажатиями. По умолчанию 0
- **inRestoreStateAfterBool** (*bool*, *опциональный*) – Может использоваться, чтобы восстановить регистр нажатых ключей после того, как текст напечатан, т.е. нажимает ключи, которые были выпущены в начало.
- **inExactBool** (*bool*, *опциональный*) – Печатает все знаки как явный unicode. Необязательный параметр
- **inWaitAfterSecFloat** (*float*, *опциональный*) – Количество секунд, которые ожидать после выполнения операции. По умолчанию установлено в настройках модуля Keyboard (базовое значение 0.4)

Коды клавиш

Ниже представлены коды горячих клавиш, а также символов русской и английской раскладки.

```
# ШЕСТНАДЦАТИРИЧНЫЙ СКАН-КОД В РУССКОЙ РАСКЛАДКЕ (НЕЗАВИСИМО ОТ ВЫБРАННОГО ЯЗЫКА НА КЛАВИАТУРЕ)
# ОТОБРАЖЕНИЕ СКАН КОДОВ НА КЛАВИАТУРЕ https://snipp.ru/handbk/scan-codes

KEY_RUS_Ф = 0x1E #A
KEY_RUS_И = 0x30 #B
KEY_RUS_С = 0x2E #C
KEY_RUS_В = 0x20 #D
KEY_RUS_У = 0x12 #E
KEY_RUS_А = 0x21 #F
KEY_RUS_П = 0x22 #G
KEY_RUS_Р = 0x23 #H
KEY_RUS_Ш = 0x17 #I
KEY_RUS_О = 0x24 #J
KEY_RUS_Л = 0x25 #K
KEY_RUS_Д = 0x26 #L
KEY_RUS_Ь = 0x32 #M
KEY_RUS_Т = 0x31 #N
KEY_RUS_Щ = 0x18 #O
KEY_RUS_З = 0x19 #P
KEY_RUS_Й = 0x10 #Q
KEY_RUS_К = 0x13 #R
KEY_RUS_Ы = 0x1F #S
KEY_RUS_Е = 0x14 #T
KEY_RUS_Г = 0x16 #U
KEY_RUS_М = 0x2F #V
KEY_RUS_Ц = 0x11 #W
KEY_RUS_Ч = 0x2D #X
KEY_RUS_Н = 0x15 #Y
KEY_RUS_Я = 0x2C #Z
KEY_RUS_Ё = 0x29 #~
KEY_RUS_Ж = 0x27 #:
KEY_RUS_Б = 0x33 #<
KEY_RUS_Ю = 0x34 #>
KEY_RUS_Х = 0x1A #[
KEY_RUS_Ъ = 0x1B #]
KEY_RUS_Э = 0x28 #'

KEY_ENG_A = 0x1E #A
KEY_ENG_B = 0x30 #B
KEY_ENG_C = 0x2E #C
KEY_ENG_D = 0x20 #D
KEY_ENG_E = 0x12 #E
KEY_ENG_F = 0x21 #F
KEY_ENG_G = 0x22 #G
KEY_ENG_H = 0x23 #H
KEY_ENG_I = 0x17 #I
KEY_ENG_J = 0x24 #J
KEY_ENG_K = 0x25 #K
KEY_ENG_L = 0x26 #L
KEY_ENG_M = 0x32 #M
KEY_ENG_N = 0x31 #N
KEY_ENG_O = 0x18 #O
KEY_ENG_P = 0x19 #P
KEY_ENG_Q = 0x10 #Q
KEY_ENG_R = 0x13 #R
KEY_ENG_S = 0x1F #S
KEY_ENG_T = 0x14 #T
KEY_ENG_U = 0x16 #U
KEY_ENG_V = 0x2F #V
KEY_ENG_W = 0x11 #W
KEY_ENG_X = 0x2D #X
KEY_ENG_Y = 0x15 #Y
KEY_ENG_Z = 0x2C #Z


KEY_HOT_NUMPAD_0 = 0x52
KEY_HOT_NUMPAD_1 = 0x4F
KEY_HOT_NUMPAD_2 = 0x50
KEY_HOT_NUMPAD_3 = 0x51
KEY_HOT_NUMPAD_4 = 0x4B
KEY_HOT_NUMPAD_5 = 0x4C
KEY_HOT_NUMPAD_6 = 0x4D
KEY_HOT_NUMPAD_7 = 0x47
KEY_HOT_NUMPAD_8 = 0x48
KEY_HOT_NUMPAD_9 = 0x49
KEY_HOT_NUMPAD_ASTERISK = 0x37 #*
KEY_HOT_NUMPAD_PLUS = 0x4E
KEY_HOT_NUMPAD_MINUS = 0x4A
KEY_HOT_NUMPAD_DELETE = 0x53
KEY_HOT_NUMPAD_SOLIDUS = 0x35 #/
KEY_HOT_NUMPAD_ENTER = 0x11c
```

```
KEY_HOT_F1 = 0x3B
KEY_HOT_F2 = 0x3C
KEY_HOT_F3 = 0x3D
KEY_HOT_F4 = 0x3E
KEY_HOT_F5 = 0x3F
KEY_HOT_F6 = 0x40
KEY_HOT_F7 = 0x41
KEY_HOT_F8 = 0x42
KEY_HOT_F9 = 0x43
KEY_HOT_F10 = 0x44
KEY_HOT_F11 = 0x57
KEY_HOT_F12 = 0x58
KEY_HOT_F13 = 0x7C
KEY_HOT_F14 = 0x7D
KEY_HOT_F15 = 0x7E
KEY_HOT_F16 = 0x7F
KEY_HOT_F17 = 0x80
KEY_HOT_F18 = 0x81
KEY_HOT_F19 = 0x82
KEY_HOT_F20 = 0x83
KEY_HOT_F21 = 0x84
KEY_HOT_F22 = 0x85
KEY_HOT_F23 = 0x86
KEY_HOT_F24 = 0x87

KEY_HOT_TILDE = 0x29 #~
KEY_HOT_COLON = 0x27 #:
KEY_HOT_PLUS = 0x0D #+
KEY_HOT_MINUS = 0x0C #-
KEY_HOT_LESS_THAN = 0x33 #< ,
KEY_HOT_GREATER_THAN = 0x34 #> .
KEY_HOT_SOLIDUS = 0x35 #/ ?
KEY_HOT_SQUARE_BRACKET_LEFT = 0x1A #[
KEY_HOT_SQUARE_BRACKET_RIGHT = 0x1B #]
KEY_HOT_APOSTROPHE = 0x28 #' "
KEY_HOT_VERTICAL_LINE = 0x2B #| \

KEY_HOT_ESC = 0x1
KEY_HOT_BACKSPACE = 0x0E
KEY_HOT_TAB = 0x0F
KEY_HOT_ENTER = 0x1C
KEY_HOT_CONTEXT_MENU = 0x15D
KEY_HOT_SHIFT_LEFT = 0x2A
KEY_HOT_SHIFT_RIGHT = 0x36
KEY_HOT_CTRL_LEFT = 0x1D
KEY_HOT_CTRL_RIGHT = 0x11D
KEY_HOT_ALT_LEFT = 0x38
KEY_HOT_ALT_RIGHT = 0x138
KEY_HOT_WIN_LEFT = 0x5B
KEY_HOT_WIN_RIGHT = 0x5C
KEY_HOT_CAPS_LOCK = 0x3A
KEY_HOT_NUM_LOCK = 0x45
KEY_HOT_SCROLL_LOCK = 0x46
KEY_HOT_END = 0x4F
KEY_HOT_HOME = 0x47
KEY_HOT_SPACE = 0x39
KEY_HOT_PAGE_UP = 0x49
KEY_HOT_PAGE_DOWN = 0x51
KEY_HOT_CLEAR = 0x4C
KEY_HOT_LEFT = 0x4B
KEY_HOT_UP = 0x48
KEY_HOT_RIGHT = 0x4D
KEY_HOT_DOWN = 0x50
KEY_HOT_PRINT_SCREEN = 0x137
KEY_HOT_INSERT = 0x52
KEY_HOT_DELETE = 0x53

KEY_HOT_0 = 0xB
KEY_HOT_1 = 0x2
KEY_HOT_2 = 0x3
KEY_HOT_3 = 0x4
KEY_HOT_4 = 0x5
KEY_HOT_5 = 0x6
KEY_HOT_6 = 0x7
KEY_HOT_7 = 0x8
KEY_HOT_8 = 0x9
KEY_HOT_9 = 0xA
```

Дополнительная функциональность

Дополнительно модуль содержит функции вспомогательной библиотеки. Ознакомиться с описанием можно Здесь

```
# Пример использования функции send
from pyOpenRPA.Robot import Keyboard
Keyboard.send(57)
```

Быстрая навигация

- Сообщество pyOpenRPA (telegram)
- Сообщество pyOpenRPA (tenchat)
- Сообщество pyOpenRPA (вконтакте)
- Презентация pyOpenRPA
- Портал pyOpenRPA
- Репозиторий pyOpenRPA

.. v1.4.0 replace:: v1.4.0

Предыдущая Следующая

5. Функции Clipboard

Модуль для взаимодействия с буфером обмена (получить значение из буфера обмена, установить значение в буфер обмена)

!ВНИМАНИЕ! ПРИ ИСПОЛЬЗОВАНИИ НА LINUX НЕОБХОДИМО ИМЕТЬ ПАКЕТ XCLIP (sudo apt-get install xclip)

Описание функций

Описание каждой функции начинается с обозначения L+,W+, что означает, что функция поддерживается в ОС Linux (L) и Windows (W)

Functions:

`Get` ()	L+,W+: Получить текстовое содержимое буфера обмена.
`Set` (inTextStr)	L+,W+: Установить текстовое содержимое в буфер обмена.

pyOpenRPA.Robot.Clipboard.Get() [исходный код]

L+,W+: Получить текстовое содержимое буфера обмена.

```
# Clipboard: Взаимодействие с буфером
from pyOpenRPA.Robot import Clipboard
lClipStr = Clipboard.Get()
```

Результат:
Текстовое содержимое буфера обмена

Тип результата:
str

pyOpenRPA.Robot.Clipboard.Set(*inTextStr: str*) [исходный код]

L+,W+: Установить текстовое содержимое в буфер обмена.

```
# Clipboard: Взаимодействие с буфером
from pyOpenRPA.Robot import Clipboard
lClipStr = Clipboard.Set(inTextStr="HELLO WORLD")
```

Параметры:
inTextStr (*str*) – Текстовое содержимое для установки в буфера обмена

Быстрая навигация

- Сообщество pyOpenRPA (telegram)
- Сообщество pyOpenRPA (tenchat)
- Сообщество pyOpenRPA (вконтакте)

- Презентация pyOpenRPA
- Портал pyOpenRPA
- Репозиторий pyOpenRPA

.. v1.4.0 replace:: v1.4.0

Предыдущая Следующая

6. Функции Mouse

Общее

Описание функций

Описание каждой функции начинается с обозначения L+,W+, что означает, что функция поддерживается в ОС Linux (L) и Windows (W)

Functions:

`Click` ([inXInt, inYInt, inClickCountInt, ...])	L+,W+: Нажатие (вниз) кнопки мыши и затем немедленно выпуск
`ClickDouble` ([inXInt, inYInt, ...])	L+,W+: Двойное нажатие левой клавиши мыши.
`Down` ([inXInt, inYInt, inButtonStr, ...])	L+,W+: Переместить указатель по координатам inXInt, inYInt, посл
`MoveTo` ([inXInt, inYInt, ...])	L+,W+: Переместить указатель мыши на позицию inXInt, inYInt за
`ScrollHorizontal` (inScrollClickCountInt[, ...])	L+,W-: Переместить указатель мыши на позицию inXInt, inYInt и в
`ScrollVertical` (inScrollClickCountInt[, ...])	L+,W+: Переместить указатель мыши на позицию inXInt, inYInt и в
`Up` ([inXInt, inYInt, inButtonStr, ...])	L+,W+: Отпустить (вверх) клавишу мыши.

pyOpenRPA.Robot.Mouse.Click(*inXInt: Optional[int] = None, inYInt: Optional[int] = None, inClickCountInt: int = 1, inIntervalSecFloat: float = 0.0, inButtonStr: str = 'left', inMoveDurationSecFloat: float = 0.0, inWaitAfterSecFloat: float = 0.4*)
[исходный код]

L+,W+: Нажатие (вниз) кнопки мыши и затем немедленно выпуск (вверх) её. Допускается следующая параметризация. Если не указаны inXInt и inYInt - клик производится по месту нахождения указателя мыши.

!ВНИМАНИЕ! Отсчет координат inXInt, inYInt начинается с левого верхнего края рабочей области (экрана).

```
# Mouse: Взаимодействие с мышью
from pyOpenRPA.Robot import Mouse
Mouse.Click(100,150) #Выполнить нажатие левой клавиши мыши на экране по координатам: X(гор) 100px, Y(в
```

Параметры:

- **inXInt** (*int*, *опциональный*) – Целевая позиция указателя мыши по оси X (горизонтальная ось).
- **inYInt** (*int*, *опциональный*) – Целевая позиция указателя мыши по оси Y (вертикальная ось).
- **inClickCountInt** (*int*, *опциональный*) – Количество нажатий (вниз и вверх) кнопкой мыши, По умолчанию 1
- **inIntervalSecFloat** (*float*, *опциональный*) – Интервал ожидания в секундах между нажатиями, По умолчанию 0.0
- **inButtonStr** (*str*, *опциональный*) – Номер кнопки, которую требуется нажать. Возможные варианты: „left“, „middle“, „right“ или 1, 2, 3. В остальных случаях инициирует исключение

ValueError. По умолчанию „left“
- **inMoveDurationSecFloat** (*float*, *опциональный*) – Время перемещения указателя мыши, По умолчанию 0.0 (моментальное перемещение)
- **inWaitAfterSecFloat** (*float*, *опциональный*) – Количество секунд, которые ожидать после выполнения операции. По умолчанию установлено в настройках модуля Mouse (базовое значение 0.4)

pyOpenRPA.Robot.Mouse.ClickDouble(*inXInt: Optional[int] = None, inYInt: Optional[int] = None, inWaitAfterSecFloat: float = 0.4*) [исходный код]

L+,W+: Двойное нажатие левой клавиши мыши. Данное действие аналогично вызову функции (см. ниже).

!ВНИМАНИЕ! Отсчет координат inXInt, inYInt начинается с левого верхнего края рабочей области (экрана).

```
# Mouse: Взаимодействие с мышью
from pyOpenRPA.Robot import Mouse
Mouse.ClickDouble(100,150) #Выполнить двойное нажатие левой клавиши мыши на экране по координатам: X(г
```

Параметры:
- **inXInt** (*int*, *опциональный*) – Целевая позиция указателя мыши по оси X (горизонтальная ось).
- **inYInt** (*int*, *опциональный*) – Целевая позиция указателя мыши по оси Y (вертикальная ось).
- **inWaitAfterSecFloat** (*float*, *опциональный*) – Количество секунд, которые ожидать после выполнения операции. По умолчанию установлено в настройках модуля Mouse (базовое значение 0.4)

pyOpenRPA.Robot.Mouse.Down(*inXInt: Optional[int] = None, inYInt: Optional[int] = None, inButtonStr: str = 'left', inWaitAfterSecFloat: float = 0.4*) [исходный код]

L+,W+: Переместить указатель по координатам inXInt, inYInt, после чего нажать (вниз) клавишу мыши и не отпускать до выполнения соответсвующей команды (см. Up). Если координаты inXInt, inYInt не переданы - нажатие происходит на тех координатах X/Y, на которых указатель мыши находится.

!ВНИМАНИЕ! Отсчет координат inXInt, inYInt начинается с левого верхнего края рабочей области (экрана).

```
# Mouse: Взаимодействие с мышью
from pyOpenRPA.Robot import Mouse
Mouse.Down() #Опустить левую клавишу мыши
```

Параметры:
- **inXInt** (*int*, *опциональный*) – Целевая позиция указателя мыши по оси X (горизонтальная ось).
- **inYInt** (*int*, *опциональный*) – Целевая позиция указателя мыши по оси Y (вертикальная ось).
- **inButtonStr** (*str*, *опциональный*) – Номер кнопки, которую требуется нажать. Возможные варианты: „left“, „middle“, „right“ или 1, 2, 3. В остальных случаях инициирует исключение ValueError. По умолчанию „left“
- **inWaitAfterSecFloat** (*float*, *опциональный*) – Количество секунд, которые ожидать после выполнения операции. По умолчанию установлено в настройках модуля Mouse (базовое значение 0.4)

pyOpenRPA.Robot.Mouse.MoveTo(*inXInt=None, inYInt=None, inMoveDurationSecFloat: float = 0.0, inWaitAfterSecFloat: float = 0.4*) [исходный код]

L+,W+: Переместить указатель мыши на позицию inXInt, inYInt за время inMoveDurationSecFloat.

!ВНИМАНИЕ! Отсчет координат inXInt, inYInt начинается с левого верхнего края рабочей области

(экрана).

```
# Mouse: Взаимодействие с мышью
from pyOpenRPA.Robot import Mouse
Mouse.MoveTo(inXInt=100, inYInt=200) #Переместить указатель мыши по координатам: X(гор) 100, Y(вер) 20
```

Параметры:

- **inXInt** (*int*, *опциональный*) – Целевая позиция указателя мыши по оси X (горизонтальная ось).
- **inYInt** (*int*, *опциональный*) – Целевая позиция указателя мыши по оси Y (вертикальная ось).
- **inMoveDurationSecFloat** (*float*, *опциональный*) – Время перемещения указателя мыши, По умолчанию 0.0 (моментальное перемещение)
- **inWaitAfterSecFloat** (*float*, *опциональный*) – Количество секунд, которые ожидать после выполнения операции. По умолчанию установлено в настройках модуля Mouse (базовое значение 0.4)

pyOpenRPA.Robot.Mouse.ScrollHorizontal(*inScrollClickCountInt*, *inXInt=None*, *inYInt=None*, *inWaitAfterSecFloat: float = 0.4*) [исходный код]

L+,W-: Переместить указатель мыши на позицию inXInt, inYInt и выполнить горизонтальную прокрутку (скроллинг) виртуальным колесом мыши на количество щелчков inScrollClickCountInt.

!ВНИМАНИЕ! Отсчет координат inXInt, inYInt начинается с левого верхнего края рабочей области (экрана).

```
# Mouse: Взаимодействие с мышью
from pyOpenRPA.Robot import Mouse
Mouse.ScrollHorizontal(100, inXInt=100, inYInt=200) #Крутить колесо мыши вниз на 100 кликов по координ
Mouse.ScrollHorizontal(-100) #Крутить колесо мыши вверх на 100 кликов по текущим координатам указателя
```

Параметры:

- **inScrollClickCountInt** (*int*, *обязательный*) – Количество щелчок колеса мыши, которое требуется !горизонтально! прокрутить. Аргумент может принимать как положительное (прокрутка вправо), так и отрицательное (прокрутка влево) значения
- **inXInt** (*int*, *опциональный*) – Целевая позиция указателя мыши по оси X (горизонтальная ось).
- **inYInt** (*int*, *опциональный*) – Целевая позиция указателя мыши по оси Y (вертикальная ось).
- **inWaitAfterSecFloat** (*float*, *опциональный*) – Количество секунд, которые ожидать после выполнения операции. По умолчанию установлено в настройках модуля Mouse (базовое значение 0.4)

pyOpenRPA.Robot.Mouse.ScrollVertical(*inScrollClickCountInt*, *inXInt=None*, *inYInt=None*, *inWaitAfterSecFloat: float = 0.4*) [исходный код]

L+,W+: Переместить указатель мыши на позицию inXInt, inYInt и выполнить вертикальную прокрутку (скроллинг) колесом мыши на количество щелчков inScrollClickCountInt.

!ВНИМАНИЕ! Отсчет координат inXInt, inYInt начинается с левого верхнего края рабочей области (экрана).

```
# Mouse: Взаимодействие с мышью
from pyOpenRPA.Robot import Mouse
Mouse.ScrollVertical(100, inXInt=100, inYInt=200) #Крутить колесо мыши вниз на 100 кликов по координат
Mouse.ScrollVertical(-100) #Крутить колесо мыши вверх на 100 кликов по текущим координатам указателя м
```

Параметры:

- **inScrollClickCountInt** (*int*, *обязательный*) – Количество щелчок колеса мыши, которое требуется !вертикально! прокрутить. Аргумент может принимать как положительное

(прокрутка вниз), так и отрицательное (прокрутка вверх) значения
- **inXInt** (*int*, *опциональный*) – Целевая позиция указателя мыши по оси X (горизонтальная ось).
- **inYInt** (*int*, *опциональный*) – Целевая позиция указателя мыши по оси Y (вертикальная ось).
- **inWaitAfterSecFloat** (*float*, *опциональный*) – Количество секунд, которые ожидать после выполнения операции. По умолчанию установлено в настройках модуля Mouse (базовое значение 0.4)

pyOpenRPA.Robot.Mouse.Up(*inXInt: Optional[int] = None, inYInt: Optional[int] = None, inButtonStr: str = 'left', inWaitAfterSecFloat: float = 0.4*) [исходный код]

L+,W+: Отпустить (вверх) клавишу мыши. Если координаты inXInt, inYInt не переданы - нажатие происходит на тех координатах X/Y, на которых указатель мыши находится.

!ВНИМАНИЕ! Отсчет координат inXInt, inYInt начинается с левого верхнего края рабочей области.

```
# Mouse: Взаимодействие с мышью
from pyOpenRPA.Robot import Mouse
Mouse.Up(inButtonStr:str='right') #Поднять правую клавишу мыши
```

Параметры:
- **inXInt** (*int*, *опциональный*) – Целевая позиция указателя мыши по оси X (горизонтальная ось).
- **inYInt** (*int*, *опциональный*) – Целевая позиция указателя мыши по оси Y (вертикальная ось).
- **inButtonStr** (*str*, *опциональный*) – Номер кнопки, которую требуется поднять. Возможные варианты: „left“, „middle“, „right“ или 1, 2, 3. В остальных случаях инициирует исключение ValueError. По умолчанию „left“
- **inWaitAfterSecFloat** (*float*, *опциональный*) – Количество секунд, которые ожидать после выполнения операции. По умолчанию установлено в настройках модуля Mouse (базовое значение 0.4)

Быстрая навигация

- Сообщество pyOpenRPA (telegram)
- Сообщество pyOpenRPA (tenchat)
- Сообщество pyOpenRPA (вконтакте)
- Презентация pyOpenRPA
- Портал pyOpenRPA
- Репозиторий pyOpenRPA

.. v1.4.0 replace:: v1.4.0

Предыдущая Следующая

7. Функции Screen

Общее

!ВНИМАНИЕ! ДЛЯ РАБОТЫ В LINUX ТРЕБУЕТСЯ НАЛИЧИЕ КОМПОНЕНТА SCROT (sudo apt-get scrot)

Дорогие коллеги!

Мы знаем, что с pyOpenRPA вы сможете существенно улучшить качество вашего бизнеса. Платформа роботизации pyOpenRPA - это разработка, которая дает возможность делать виртуальных сотрудников (программных роботов RPA) выгодными, начиная от эффекта всего в **10 тыс. руб.** И управлять ими будете только Вы!

Если у вас останутся вопросы, то вы всегда можете обратиться в центр поддержки клиентов pyOpenRPA. Контакты: 2. Лицензия & Контакты

pyOpenRPA - роботы помогут!

Класс Box

Экземпляр класса pyscreeze.Box, который характеризует прямоугольную область на экране.

top - координата левой верхней точки в пикселях по оси X (горизонталь) left - координата левой верхней точки в пикселях по оси Y (вертикаль) height - расстояние вниз от левой верхней точки в пикселях width - расстояние вправо от левой верхней точки в пикселях

Класс Point

Экземпляр класса pyscreeze.Point, который характеризует точку на экране.

x - координата точки в пикселях по оси X (горизонталь) y - координата точки в пикселях по оси Y (вертикаль)

Символьное указание точки (inPointRuleStr)

LU|CU|RU

LC|CC|RC

LD|CD|RD

Символьное указание точки - точка относительно которой выполнить изменение прямоугольной области.

«CC»,

Формат образования кода: XY, где

- X обозначает положение по горизонтали (допустимые значения: «L», «C», «R»)
- Y обозначает положение по вертикали (допустимые значения: «U», «C», «D»)

Допустимые значения:

- «CC» - центр по горизонтали, центр по вертикали
- «LU» - левый край по горизонтали, верхний край по вертикали
- «LD» - левый край по горизонтали, нижний край по вертикали
- «RD» - правый край по горизонтали, нижний край по вертикали
- «RU» - правый край по горизонтали, верхний край по вертикали

X-10 - корректировка координаты по оси X на 10 пикселей влево Y+20 - корректировка координаты по оси Y на 20 пикселей вниз

Символьное указание области (inAnchorRuleStr)

LU|CU|RU

LC|CC|RC

LD|CD|RD

Символьное указание области поиска - область относительно которой выполнить поиск другой прямоугольной области.

«CC»,

Формат образования кода: XY, где

- X обозначает область по горизонтали (допустимые значения: «L», «C», «R»)
- Y обозначает область по вертикали (допустимые значения: «U», «C», «D»)

Допустимые значения:

- «CC» - выбранная прямоугольная область
- «LU» - слева и сверху от выбранной прямоугольной области
- «LD» - слева и снизу от выбранной прямоугольной области
- «LC» - слева от выбранной прямоугольной области
- «RC» - справа от выбранной прямоугольной области
- «CU» - сверху от выбранной прямоугольной области
- «CD» - сверху от выбранной прямоугольной области
- «RD» - справа и снизу от выбранной прямоугольной области
- «RU» - справа и сверху от выбранной прямоугольной области

Опция «S» (strict) - искомый объект должен всеми своими координатами находиться в обозначенной прямоугольной области

Формат допускает комбинации нескольких областей в одной строке. Пример: «CC,LU,LD,S» «CC|LU|LD|S» «CCLULDS»

Графическая интерпретация: +|-|- ----- -|+|- —– +|-|-

Описание функций

Описание каждой функции начинается с обозначения L+,W+, что означает, что функция поддерживается в ОС Linux (L) и Windows (W)

Functions:

`BoxAnchorRuleCheck` (inBox[, inAnchorBox, ...])	L+,W+: Выполнить проверку соответствия всем условиям вхо
`BoxCreate` (inTopInt, inLeftInt, inHeightInt, ...)	L+,W+: Создать экземпляр прямоугольной области.
`BoxDraw` (inBox[, inColorStr, inThicknessInt])	L+,W+: Выполнить подсветку прямоугольной области inBox н
`BoxGetPoint` (inBox[, inPointRuleStr])	L+,W+:Получить точку pyscreeze.Point по заданной прямоугол
`BoxModify` (inBox[, inDWidthInt, ...])	L+,W+: Изменить ширину / высоту прямоугольной области.
`BoxMoveTo` (inBox[, inDXInt, inDYInt])	L+,W+: Переместить прямоугольную область (сохранить дли
`BoxOverlay` (inBox1, inBox2)	L+,W+:Проверить наложение 2-х прямоугольных областей др
`ImageClick` (inImgPathStr[, inBoxIndexInt, ...])	L+,W+:Выполнить поиск прямоугольной области по изображ
`ImageExists` (inImgPathStr[, ...])	L+,W+:Проверить, имеется ли на экране хотя бы один подход
`ImageLocateAll` (inImgPathStr[, ...])	L+W+: Искать на экране графические объекты, которые похо
`ImageLocateAllInfo` (inImgPathStr[, ...])	L+W+: Искать на экране графические объекты, которые похо
`ImageWaitAppear` (inImgPathStr[, ...])	L+,W+:Ожидать появление изображения на протяжении inW
`ImageWaitDisappear` (inImgPathStr[, ...])	L+,W+:Ожидать исчезновение изображения на протяжении in
`InitSnipingTool` (inPath)	L-,W+:Выполнить выделение прямоугольной области на экра
`PointClick` (inPoint[, inClickCountInt, ...])	L+,W+:Нажатие (вниз) кнопки мыши и затем немедленно выг
`PointClickDouble` (inPoint[, inWaitAfterSecFloat])	L+,W+:Двойное нажатие левой клавиши мыши.
`PointCreate` (inXInt, inYInt)	L+,W+:Создать точку pyscreeze.Point.
`PointDown` (inPoint[, inButtonStr, ...])	L+,W+:Переместить указатель по координатам inPoint, после
`PointModify` (inPoint, inDXInt, inDYInt)	L+,W+:Скорректировать точку pyscreeze.Point.
`PointMoveTo` (inPoint[, inWaitAfterSecFloat])	L+,W+:Переместить указатель мыши на позицию inXInt, inYIn
`PointUp` (inPoint[, inButtonStr, ...])	L+,W+:Отпустить (вверх) клавишу мыши.
`ResolutionsGet` ()	L-,W+:Получить разрешение всех используемых экранов.

Classes:

`SnipingTool` (parent, path)	

pyOpenRPA.Robot.Screen.BoxAnchorRuleCheck(*inBox*, *inAnchorBox=None*, *inAnchorRuleStr=None*)→ bool
[исходный код]

L+,W+: Выполнить проверку соответствия всем условиям вхождения inBox в inAnchorBox с учетом правил inAnchorRule

```
# Screen: Взаимодействие с экраном
from pyOpenRPA.Robot import Screen
lBox1 = Screen.BoxCreate(inTopInt=265, inLeftInt=62, inHeightInt=100, inWidthInt=90)
lBox2 = Screen.BoxCreate(inTopInt=160, inLeftInt=160, inHeightInt=100, inWidthInt=100)
lBox3 = Screen.BoxCreate(inTopInt=460, inLeftInt=60, inHeightInt=100, inWidthInt=100)

l = Screen.BoxAnchorRuleCheck(inBox=lBox1, inAnchorBox=[lBox2,lBox3], inAnchorRuleStr=["LD","CUS"])
Screen.BoxDraw([lBox1,lBox2,lBox3])
```

Параметры:

- **inBox** (*pyscreeze.Box*) – Экземпляр класса прямоугольной области Box
- **inAnchorBox** (*pyscreeze.Box или list из pyscreeze.Box*) – Экземпляр класса прямоугольной области Box
- **inAnchorRuleStr** (*str или list из str*) – Символьное указание области проверки соответствия. Может принимать единственное значение (единый формат для всех inAnchorBox), или список форматов для каждого inAnchorBox (если inAnchorBox является списком Box)

Результат:

True - соответствует всем правилам

Тип результата:

bool

pyOpenRPA.Robot.Screen.BoxCreate(*inTopInt: int, inLeftInt: int, inHeightInt: int, inWidthInt: int*)→ Box

[исходный код]

L+,W+: Создать экземпляр прямоугольной области.

!ВНИМАНИЕ! Координаты inTopInt, inLeftInt определяют верхний левый край прямоугольной области.

Параметры:

- **inTopInt** (*int*) – Координата левой верхней точки в пикселях по оси X (горизонталь)
- **inLeftInt** (*int*) – Координата левой верхней точки в пикселях по оси Y (вертикаль)
- **inHeightInt** (*int*) – Расстояние вниз от левой верхней точки в пикселях
- **inWidthInt** (*int*) – Расстояние вправо от левой верхней точки в пикселях

pyOpenRPA.Robot.Screen.BoxDraw(*inBox, inColorStr='green', inThicknessInt=2*)

[исходный код]

L+,W+: Выполнить подсветку прямоугольной области inBox на экране

!ВНИМАНИЕ! ЦВЕТ inColorStr ПОДДЕРЖИВАЕТСЯ ТОЛЬКО НА ОС WINDOWS

!ВНИМАНИЕ! ПОДДЕРЖИВАЕТ ПАКЕТНУЮ ОБРАТКУ ПРИ ПЕРЕДАЧЕ СПИСКА ЭКЗЕМПЛЯРОВ BOX

```
# Screen: Взаимодействие с экраном
from pyOpenRPA.Robot import Screen
# ВАРИАНТ ОТРИСОВКИ 1ГО ЭЛЕМЕНТА
# Создать пробную прямоугольную область
lBox = Screen.BoxCreate(inTopInt=10, inLeftInt=10, inHeightInt=10, inWidthInt=10)
Screen.BoxDraw(lBox)

# ВАРИАНТ ПАКЕТНОЙ ОТРИСОВКИ
# Создать пробную прямоугольную область
lBox = Screen.BoxCreate(inTopInt=10, inLeftInt=10, inHeightInt=100, inWidthInt=100)
lBox2 = Screen.BoxCreate(inTopInt=60, inLeftInt=60, inHeightInt=100, inWidthInt=100)
Screen.BoxDraw([lBox, lBox2])
```

Параметры:

- **inBox** (*pyscreeze.Box*) – Экземпляр класса прямоугольной области Box
- **inColorStr** (*str, необязательный*) – цвет подсветки прямоугольной области. Варианты: „red“, „green“, „blue“. По умолчанию „green“

- **inThicknessInt** (*int*, *необязательный*) – толщина подсветки прямоугольной области. По умолчанию 2

pyOpenRPA.Robot.Screen.BoxGetPoint(*inBox*, *inPointRuleStr='CC'*)→ Point [исходный код]

L+,W+:Получить точку pyscreeze.Point по заданной прямоугольной области pyscreeze.Box и строковому параметру расположения inPointRuleStr.

```
# Screen: Взаимодействие с мышью объектами экрана
from pyOpenRPA.Robot import Screen
lBox1 = Screen.BoxCreate(inTopInt=10, inLeftInt=10, inHeightInt=100, inWidthInt=1000)
lPoint = Screen.BoxGetPoint(inBox=lBox1, inPointRuleStr="LC")
```

Параметры:

- **inBox** (*pyscreeze.Box*, *обязательный*) – Прямоугольная область на экране
- **inPointRuleStr** (*str*, *опциональный*) – Правило идентификации точки на прямоугольной области (правила формирования см. выше). Варианты: «LU»,»CU»,»RU»,»LC»,»CC»,»RC»,»LD»,»CD»,»RD». По умолчанию «CC»

Результат:

Точка на экране

Тип результата:

pyscreeze.Point

pyOpenRPA.Robot.Screen.BoxModify(*inBox*, *inDWidthInt=None*, *inDHeightInt=None*, *inPointRuleStr='CC'*) [исходный код]

L+,W+: Изменить ширину / высоту прямоугольной области.

!ВНИМАНИЕ! ПОДДЕРЖИВАЕТ ПАКЕТНУЮ ОБРАТКУ ПРИ ПЕРЕДАЧЕ СПИСКА ЭКЗЕМПЛЯРОВ BOX

!ВНИМАНИЕ! ЕСЛИ СМЕЩЕНИЕ ПРИВЕДЕТ К ОБРАЗОВАНИЮ ДРОБНОГО ЧИСЛА, БУДЕТ ВЫПОЛНЕНО ОКРУГЛЕНИЕ ПО МАТЕМАТИЧЕСКИМ ПРАВИЛАМ

```
# Screen: Взаимодействие с экраном
from pyOpenRPA.Robot import Screen
# Вариант изменения 1-го элемента
# Создать пробную прямоугольную область
lBox = Screen.BoxCreate(inTopInt=10, inLeftInt=10, inHeightInt=10, inWidthInt=10)
# Скорректировать пробную прямоугольную область
lBox2 = Screen.BoxModify(lBox,10,10,"CC"); print(lBox2)
lBox2 = Screen.BoxModify(lBox,10,10,"LU"); print(lBox2)
lBox2 = Screen.BoxModify(lBox,10,10,"LD"); print(lBox2)
lBox2 = Screen.BoxModify(lBox,10,10,"RU"); print(lBox2)
lBox2 = Screen.BoxModify(lBox,10,10,"RD"); print(lBox2)
```

Параметры:

- **inBox** (*pyscreeze.Box*) – Экземпляр класса прямоугольной области Box
- **inDXInt** (*int*, *опциональный*) – Смещение левой верхней координаты по оси X в пикселях (горизонтальная ось).
- **inDYInt** (*int*, *опциональный*) – Смещение левой верхней координаты по оси Y в пикселях (вертикальная ось).
- **inPointRuleStr** (*str*, *опциональный*) – Символьное указание точки (подробнее см. выше), относительно которой выполнить изменение прямоугольной области. Допустимые значения: «CC» (по умолчанию), «LU», «LD», «RD», «RU»

Результат:

Экземпляр класса прямоугольной области Box

Тип результата:
pyscreeze.Box

pyOpenRPA.Robot.Screen.BoxMoveTo(*inBox, inDXInt=None, inDYInt=None*) [исходный код]

L+,W+: Переместить прямоугольную область (сохранить длину/ширину).

!ВНИМАНИЕ! ПОДДЕРЖИВАЕТ ПАКЕТНУЮ ОБРАТКУ ПРИ ПЕРЕДАЧЕ СПИСКА ЭКЗЕМПЛЯРОВ BOX

```
# Screen: Взаимодействие с экраном
from pyOpenRPA.Robot import Screen
# Вариант изменения 1-го элемента
# Создать пробную прямоугольную область
lBox = Screen.BoxCreate(inTopInt=10, inLeftInt=10, inHeightInt=10, inWidthInt=10)
# Переместить пробную прямоугольную область
lBox = Screen.BoxMoveTo(lBox, inDXInt=100, inDYInt=200)
```

Параметры:

- **inBox** (*pyscreeze.Box*) – Экземпляр класса прямоугольной области Box
- **inDXInt** (*int*, *опциональный*) – Смещение левой верхней координаты по оси X в пикселях (горизонтальная ось).
- **inDYInt** (*int*, *опциональный*) – Смещение левой верхней координаты по оси Y в пикселях (вертикальная ось).

Результат:
Экземпляр класса прямоугольной области Box

Тип результата:
pyscreeze.Box

pyOpenRPA.Robot.Screen.BoxOverlay(*inBox1, inBox2*)→ bool [исходный код]

L+,W+:Проверить наложение 2-х прямоугольных областей друг на друга.

```
# Screen: Взаимодействие с экраном
from pyOpenRPA.Robot import Screen
lBox1 = Screen.BoxCreate(inTopInt=10, inLeftInt=10, inHeightInt=100, inWidthInt=1000)
lBox2 = Screen.BoxCreate(inTopInt=160, inLeftInt=160, inHeightInt=100, inWidthInt=100)
Screen.BoxDraw([lBox1, lBox2])
Screen.BoxOverlay(lBox1,lBox2)
```

Параметры:

- **inBox1** (*pyscreeze.Box*) – Экземпляр класса прямоугольной области Box
- **inBox2** (*pyscreeze.Box*) – Экземпляр класса прямоугольной области Box

Результат:
True - inBox1 наложен на inBox2

Тип результата:
bool

pyOpenRPA.Robot.Screen.ImageClick(*inImgPathStr: str, inBoxIndexInt: int = 0, inPointRuleStr: str = 'CC', inIsGrayModeBool: bool = False, inConfidenceFloat: Optional[float] = None, inWaitSecFloat: float = 0, inWaitIntervalSecFloat: float = 0*) [исходный код]

L+,W+:Выполнить поиск прямоугольной области по изображению.

!ВНИМАНИЕ! Для использования параметра точности inConfidenceFloat необходим пакет Python opencv-python (python -m pip install opencv-python)

```
# Screen: Взаимодействие с объектами экрана
from pyOpenRPA.Robot import Screen
Screen.ImageClick(inImgPathStr="Button.png",inConfidenceFloat=0.9)
```

Параметры:

- **inImgPathStr** (*str*, *относительный или абсолютный*) – Путь к изображению, которое требуется искать на экране
- **inBoxIndexInt** (*int*, *опционально*) – Индекс прямоугольной области, по которой выполнить клик (если обнаружено несколько областей Box), По умолчанию 0
- **inPointRuleStr** (*str*, *опциональный*) – Правило идентификации точки на прямоугольной области (правила формирования см. выше). Варианты: «LU»,»CU»,»RU»,»LC»,»CC»,»RC»,»LD»,»CD»,»RD». По умолчанию «CC»
- **inIsGrayModeBool** (*bool*, *опционально*) – True - выполнить поиск изображения в режиме серых оттенков (ускоряет производительность, если допускается искажение цвета). По умолчанию False
- **inConfidenceFloat** (*float*, *опционально*) – Показатель точности. 1.0 - идентичное соответствие, 0.0 - полное несоответствие. По умолчанию 1.0 (None)
- **inWaitSecFloat** (*float*, *опциональный*) – Время ожидания появления изображения в сек. По умолчанию 0
- **inWaitIntervalSecFloat** (*float*, *опциональный*) – Интервал повторной проверки наличия изображения. По умолчанию 0

pyOpenRPA.Robot.Screen.ImageExists(*inImgPathStr: str*, *inIsGrayModeBool: bool = False*, *inConfidenceFloat: Optional[float] = None*)→ list [исходный код]

L+,W+:Проверить, имеется ли на экране хотя бы один подходящий объект. Вернуть булево значение

!ВНИМАНИЕ! Для использования параметра точности inConfidenceFloat необходим пакет Python opencv-python (python -m pip install opencv-python)

```
# Screen: Взаимодействие с объектами экрана
from pyOpenRPA.Robot import Screen
lResult = Screen.ImageExists(inImgPathStr="Button.png",inConfidenceFloat=0.9)
```

Параметры:

- **inImgPathStr** (*str*, *относительный или абсолютный*) – Путь к изображению, которое требуется искать на экране
- **inIsGrayModeBool** (*bool*, *опционально*) – True - выполнить поиск изображения в режиме серых оттенков (ускоряет производительность, если допускается искажение цвета). По умолчанию False
- **inConfidenceFloat** (*float*, *опционально*) – Показатель точности. 1.0 - идентичное соответствие, 0.0 - полное несоответствие. По умолчанию 1.0 (None)

Результат:

Список из pyscreeze.Box

Тип результата:

list

pyOpenRPA.Robot.Screen.ImageLocateAll(*inImgPathStr: str*, *inIsGrayModeBool: bool = False*, *inConfidenceFloat: Optional[float] = None*)→ list [исходный код]

L+W+: Искать на экране графические объекты, которые похожи на inImgPathStr. Вернуть список прямоугольных областей на экране (pyscreeze.Box)

!ВНИМАНИЕ! Для использования параметра точности inConfidenceFloat необходим пакет Python

opencv-python (python -m pip install opencv-python)

```
# Screen: Взаимодействие с объектами экрана
from pyOpenRPA.Robot import Screen
Screen.ImageLocateAll(inImgPathStr="Button.png",inConfidenceFloat=0.9)
```

Параметры:

- **inImgPathStr** (*str*, *относительный или абсолютный*) – Путь к изображению, которое требуется искать на экране
- **inIsGrayModeBool** (*bool*, *опционально*) – True - выполнить поиск изображения в режиме серых оттенков (ускоряет производительность, если допускается искажение цвета). По умолчанию False
- **inConfidenceFloat** (*float*, *опционально*) – Показатель точности. 1.0 - идентичное соответствие, 0.0 - полное несоответствие. По умолчанию 1.0 (None)

Результат:
Список из pyscreeze.Box

Тип результата:
list

pyOpenRPA.Robot.Screen.ImageLocateAllInfo(*inImgPathStr: str, inIsGrayModeBool: bool = False, inConfidenceFloat: Optional[float] = None*)→ list [исходный код]

L+W+: Искать на экране графические объекты, которые похожи на inImgPathStr. Вернуть список прямоугольных областей на экране (pyscreeze.Box)

!ВНИМАНИЕ! Для использования параметра точности inConfidenceFloat необходим пакет Python opencv-python (python -m pip install opencv-python)

```
# Screen: Взаимодействие с объектами экрана
from pyOpenRPA.Robot import Screen
Screen.ImageLocateAll(inImgPathStr="Button.png",inConfidenceFloat=0.9)
```

Параметры:

- **inImgPathStr** (*str*, *относительный или абсолютный*) – Путь к изображению, которое требуется искать на экране
- **inIsGrayModeBool** (*bool*, *опционально*) – True - выполнить поиск изображения в режиме серых оттенков (ускоряет производительность, если допускается искажение цвета). По умолчанию False
- **inConfidenceFloat** (*float*, *опционально*) – Показатель точности. 1.0 - идентичное соответствие, 0.0 - полное несоответствие. По умолчанию 1.0 (None)

Результат:
Список из { left: 777, top: 497, width: 264, height: 52 }

Тип результата:
list

pyOpenRPA.Robot.Screen.ImageWaitAppear(*inImgPathStr: str, inWaitSecFloat: float = 60, inWaitIntervalSecFloat: float = 1.0, inIsGrayModeBool: bool = False, inConfidenceFloat: Optional[float] = None*)→ list [исходный код]

L+,W+:Ожидать появление изображения на протяжении inWaitSecFloat секунд. Проверять с периодичностью inWaitIntervalSecFloat. Вернуть список прямоугольных областей, которые удовлетворяют условию

!ВНИМАНИЕ! Для использования параметра точности inConfidenceFloat необходим пакет Python opencv-python (python -m pip install opencv-python)

```
# Screen: Взаимодействие с объектами экрана
from pyOpenRPA.Robot import Screen
lBoxList = Screen.ImageWaitAppear(inImgPathStr="Button.png",inConfidenceFloat=0.9)
```

Параметры:

- **inImgPathStr** (*str*, *относительный или абсолютный*) – Путь к изображению, которое требуется искать на экране
- **inWaitSecFloat** (*float*, *опциональный*) – Время ожидания появления изображения в сек. По умолчанию IMAGE_WAIT_SEC_FLOAT (60)
- **inWaitIntervalSecFloat** (*float*, *опциональный*) – Интервал повторной проверки наличия изображения. По умолчанию IMAGE_WAIT_INTERVAL_SEC_FLOAT (1)
- **inIsGrayModeBool** (*bool*, *опционально*) – True - выполнить поиск изображения в режиме серых оттенков (ускоряет производительность, если допускается искажение цвета). По умолчанию False
- **inConfidenceFloat** (*float*, *опционально*) – Показатель точности. 1.0 - идентичное соответствие, 0.0 - полное несоответствие. По умолчанию 1.0 (None)

Результат:

Список из pyscreeze.Box или [] если прошло время ожидания.

Тип результата:

list

pyOpenRPA.Robot.Screen.ImageWaitDisappear(*inImgPathStr: str, inWaitSecFloat: float = 60, inWaitIntervalSecFloat: float = 1.0, inIsGrayModeBool: bool = False, inConfidenceFloat: Optional[float] = None*) [исходный код]

L+,W+:Ожидать исчезновение изображения на протяжении inWaitSecFloat секунд. Проверять с периодичностью inWaitIntervalSecFloat.

!ВНИМАНИЕ! Для использования параметра точности inConfidenceFloat необходим пакет Python opencv-python (python -m pip install opencv-python)

```
# Screen: Взаимодействие с объектами экрана
from pyOpenRPA.Robot import Screen
Screen.ImageWaitDisappear(inImgPathStr="Button.png",inConfidenceFloat=0.9)
```

Параметры:

- **inImgPathStr** (*str*, *относительный или абсолютный*) – Путь к изображению, которое требуется искать на экране
- **inWaitSecFloat** (*float*, *опциональный*) – Время ожидания появления изображения в сек. По умолчанию IMAGE_WAIT_SEC_FLOAT (60)
- **inWaitIntervalSecFloat** (*float*, *опциональный*) – Интервал повторной проверки наличия изображения. По умолчанию IMAGE_WAIT_INTERVAL_SEC_FLOAT (1)
- **inIsGrayModeBool** (*bool*, *опционально*) – True - выполнить поиск изображения в режиме серых оттенков (ускоряет производительность, если допускается искажение цвета). По умолчанию False
- **inConfidenceFloat** (*float*, *опционально*) – Показатель точности. 1.0 - идентичное соответствие, 0.0 - полное несоответствие. По умолчанию 1.0 (None)

pyOpenRPA.Robot.Screen.InitSnipingTool(*inPath*) [исходный код]

L-,W+:Выполнить выделение прямоугольной области на экране и сохраненить его как изображение.

!ВНИМАНИЕ! Для того, чтобы временно свернуть окно, нажмите кнопку CTRL, чтобы раскрыть - отпустите кнопку CTRL

```
# Screen: Взаимодействие с объектами экрана
from pyOpenRPA.Robot import Screen
Screen.InitSnipingTool(inPath="Screenshot.png")
```

Параметры:

inPath (*str, относительный или абсолютный*) – Путь, по которому будет сохранена выделенная область

pyOpenRPA.Robot.Screen.PointClick(*inPoint: Point, inClickCountInt: int = 1, inIntervalSecFloat: float = 0.0, inButtonStr: str = 'left', inMoveDurationSecFloat: float = 0.0, inWaitAfterSecFloat: Optional[float] = None*) [исходный код]

L+,W+:Нажатие (вниз) кнопки мыши и затем немедленно выпуск (вверх) её. Допускается следующая параметризация.

```
# Screen: Взаимодействие с мышью объектами экрана
from pyOpenRPA.Robot import Screen
lPoint = Screen.PointCreate(100,150)
Screen.PointClick(lPoint) #Выполнить нажатие левой клавиши мыши
```

Параметры:

- **inPoint** (*pyscreeze.Point*, *обязательный*) – Точка на экране, по которой выполнить нажатие мыши
- **inClickCountInt** (*int*, *опциональный*) – Количество нажатий (вниз и вверх) кнопкой мыши, По умолчанию 1
- **inIntervalSecFloat** (*float*, *опциональный*) – Интервал ожидания в секундах между нажатиями, По умолчанию 0.0
- **inButtonStr** (*str*, *опциональный*) – Номер кнопки, которую требуется нажать. Возможные варианты: „left“, „middle“, „right“ или 1, 2, 3. В остальных случаях инициирует исключение ValueError. По умолчанию „left“
- **inMoveDurationSecFloat** (*float*, *опциональный*) – Время перемещения указателя мыши, По умолчанию 0.0 (моментальное перемещение)
- **inWaitAfterSecFloat** (*float*, *опциональный*) – Количество секунд, которые ожидать после выполнения операции. По умолчанию установлено в настройках модуля Mouse (базовое значение 0.4)

pyOpenRPA.Robot.Screen.PointClickDouble(*inPoint: Point, inWaitAfterSecFloat: Optional[float] = None*) [исходный код]

L+,W+:Двойное нажатие левой клавиши мыши. Данное действие аналогично вызову функции (см. ниже).

```
# Screen: Взаимодействие с мышью объектами экрана
from pyOpenRPA.Robot import Screen
lPoint = Screen.PointCreate(100,150)
Screen.PointClickDouble(lPoint) #Выполнить двойное нажатие левой клавиши мыши
```

Параметры:

- **inPoint** (*pyscreeze.Point*, *обязательный*) – Точка на экране, по которой выполнить нажатие мыши
- **inWaitAfterSecFloat** (*float*, *опциональный*) – Количество секунд, которые ожидать после выполнения операции. По умолчанию установлено в настройках модуля Mouse (базовое значение 0.4)

pyOpenRPA.Robot.Screen.PointCreate(*inXInt, inYInt*) [исходный код]

L+,W+:Создать точку pyscreeze.Point.

```
# Screen: Взаимодействие с мышью объектами экрана
from pyOpenRPA.Robot import Screen
lPoint = Screen.PointCreate(inXInt=10, inYInt=10)
```

Параметры:

- **inXInt** (*int*, *опциональный*) – Смещение указателя мыши по оси X (горизонтальная ось).
- **inYInt** (*int*, *опциональный*) – Смещение указателя мыши по оси Y (вертикальная ось).

Результат:

Точка на экране

Тип результата:

pyscreeze.Point

pyOpenRPA.Robot.Screen.PointDown(*inPoint: Point, inButtonStr: str = 'left', inWaitAfterSecFloat: Optional[float] = None*) [исходный код]

L+,W+:Переместить указатель по координатам inPoint, после чего нажать (вниз) клавишу мыши и не отпускать до выполнения соответсвующей команды (см. Up).

```
# Screen: Взаимодействие с мышью объектами экрана
from pyOpenRPA.Robot import Screen
lPoint = Screen.PointCreate(100,150)
Screen.PointDown(lPoint)
```

Параметры:

- **inPoint** (*pyscreeze.Point*, *обязательный*) – Точка на экране, по которой выполнить нажатие мыши
- **inButtonStr** (*str*, *опциональный*) – Номер кнопки, которую требуется нажать. Возможные варианты: „left“, „middle“, „right“ или 1, 2, 3. В остальных случаях инициирует исключение ValueError. По умолчанию „left“
- **inWaitAfterSecFloat** (*float*, *опциональный*) – Количество секунд, которые ожидать после выполнения операции. По умолчанию установлено в настройках модуля Mouse (базовое значение 0.4)

pyOpenRPA.Robot.Screen.PointModify(*inPoint, inDXInt, inDYInt*)→ Point [исходный код]

L+,W+:Скорректировать точку pyscreeze.Point.

```
# Screen: Взаимодействие с мышью объектами экрана
from pyOpenRPA.Robot import Screen
lPoint = Screen.PointCreate(inXInt=10, inYInt=10)
lPoint = Screen.PointModify(inPoint=lPoint, inDXInt=90, inDYInt=10)
```

Параметры:

- **inPoint** (*pyscreeze.Point*, *обязательный*) – Точка на экране, по которой выполнить нажатие мыши
- **inDXInt** (*int*, *опциональный*) – Смещение указателя мыши по оси X (горизонтальная ось).
- **inDYInt** (*int*, *опциональный*) – Смещение указателя мыши по оси Y (вертикальная ось).

Результат:

Точка на экране

Тип результата:

pyscreeze.Point

pyOpenRPA.Robot.Screen.PointMoveTo(*inPoint: Point, inWaitAfterSecFloat: Optional[float] = None*) [исходный код]

L+,W+:Переместить указатель мыши на позицию inXInt, inYInt за время inMoveDurationSecFloat.

!ВНИМАНИЕ! Отсчет координат inXInt, inYInt начинается с левого верхнего края рабочей области (экрана).

```
# Screen: Взаимодействие с мышью объектами экрана
from pyOpenRPA.Robot import Screen
lPoint = Screen.PointCreate(100,150)
Screen.PointMoveTo(inXInt=100, inYInt=200)
```

Параметры:

- **inPoint** (*pyscreeze.Point*, *обязательный*) – Точка на экране, по которой выполнить нажатие мыши
- **inWaitAfterSecFloat** (*float*, *опциональный*) – Количество секунд, которые ожидать после выполнения операции. По умолчанию установлено в настройках модуля Mouse (базовое значение 0.4)

pyOpenRPA.Robot.Screen.PointUp(*inPoint: Point, inButtonStr: str = 'left', inWaitAfterSecFloat: Optional[float] = None*) [исходный код]

L+,W+:Отпустить (вверх) клавишу мыши.

```
# Screen: Взаимодействие с мышью объектами экрана
from pyOpenRPA.Robot import Screen
lPoint = Screen.PointCreate(100,150)
Screen.PointUp(lPoint)
```

Параметры:

- **inPoint** (*pyscreeze.Point*, *обязательный*) – Точка на экране, по которой выполнить нажатие мыши
- **inButtonStr** (*str*, *опциональный*) – Номер кнопки, которую требуется поднять. Возможные варианты: „left“, „middle“, „right“ или 1, 2, 3. В остальных случаях инициирует исключение ValueError. По умолчанию „left“
- **inWaitAfterSecFloat** (*float*, *опциональный*) – Количество секунд, которые ожидать после выполнения операции. По умолчанию установлено в настройках модуля Mouse (базовое значение 0.4)

pyOpenRPA.Robot.Screen.ResolutionsGet() [исходный код]

L-,W+:Получить разрешение всех используемых экранов.

!ВНИМАНИЕ! Данная функция возвращает разрешения с учетом scale factor (масштабирование в настройках Windows)

```
# Screen: Взаимодействие с объектами экрана
from pyOpenRPA.Robot import Screen
Screen.ImageClick(inImgPathStr="Button.png",inConfidenceFloat=0.9)
```

Результат:

Список с разрешением в формате [ширина, высота].

Тип результата:

list

class pyOpenRPA.Robot.Screen.SnipingTool(*parent*, *path*) [исходный код]

Methods:

`getMonitorsResolution` ()	L-,W+: Определение разрешения всей области экрана (охватывает все подключе
`movingRect` (event)	L-,W+: Построения выделяемой прямоугольной области
`startRect` (event)	L-,W+: Начало построения выделяемой прямоугольной области
`stopRect` (event)	L-,W+: Завершение построения выделяемой прямоугольной области

getMonitorsResolution() [исходный код]

L-,W+: Определение разрешения всей области экрана (охватывает все подключенные мониторы)

movingRect(*event*) [исходный код]

L-,W+: Построения выделяемой прямоугольной области

Параметры:
event (*событие*) – Перемешение мыши

startRect(*event*) [исходный код]

L-,W+: Начало построения выделяемой прямоугольной области

Параметры:
event (*событие*) – Нажатие клавиши мыши

stopRect(*event*) [исходный код]

L-,W+: Завершение построения выделяемой прямоугольной области

Параметры:
event (*событие*) – Поднятие клавиши мыши

Быстрая навигация

- Сообщество pyOpenRPA (telegram)
- Сообщество pyOpenRPA (tenchat)
- Сообщество pyOpenRPA (вконтакте)
- Презентация pyOpenRPA
- Портал pyOpenRPA
- Репозиторий pyOpenRPA

.. v1.4.0 replace:: v1.4.0

Предыдущая Следующая

8. Функции Audio

!ВНИМАНИЕ! ДЛЯ КОРРЕКТНОЙ РАБОТЫ МОДУЛЯ ТРЕБУЕТСЯ КОМПОНЕНТ ffmpeg.exe (Для Windows x64 можно найти в pyOpenRPAResourcesWAudio). На него должен указывать один из путей в переменной окружения PATH. Или ffmpeg.exe должен быть расположен в рабочей директории (working directory)

!ВНИМАНИЕ! ДЛЯ ВИРТУАЛЬНОЙ МАШИНЫ МОЖЕТ ПОТРЕБОВАТЬСЯ КОМПОНЕНТ ВИРТУАЛЬНОГО АУДИОДРАЙВЕРА (Для Windows x64 можно найти в pyOpenRPAResourcesWAudioVBCABLE_Driver_Pack43.zip)

Общее

Дорогие коллеги!

Мы знаем, что с pyOpenRPA вы сможете существенно улучшить качество вашего бизнеса. Платформа роботизации pyOpenRPA - это разработка, которая дает возможность делать виртуальных сотрудников (программных роботов RPA) выгодными, начиная от эффекта всего в **10 тыс. руб.** И управлять ими будете только Вы!

Если у вас останутся вопросы, то вы всегда можете обратиться в центр поддержки клиентов pyOpenRPA. Контакты: 2. Лицензия & Контакты

pyOpenRPA - роботы помогут!

Класс Recorder

Экземпляр класса pyOpenRPA.Robot.Audio.Recorder, который обеспечивает захват звука (с микрофона или из приложений) и сохраняет в виде аудиофайла (множества аудиофайлов)

Описание функций

Описание каждой функции начинается с обозначения L-,W+, что означает, что функция не поддерживается в ОС Linux (L) и поддерживается в Windows (W)

Functions:

`DeviceListGet` ()	L-,W+: Вернуть список аудио устройст (входящих и исходящих, микрофонов и дина
`DeviceMicrophoneIndex` ()	L-,W+: Выполнить поиск устройства, с помощью которого можно будет выполнить
`DeviceSystemSoundIndex` ()	L-,W+: Выполнить поиск устройства, с помощью которого можно будет выполнить

pyOpenRPA.Robot.Audio.DeviceListGet() [исходный код]

L-,W+: Вернуть список аудио устройст (входящих и исходящих, микрофонов и динамиков).

from pyOpenRPA.Robot import Audio Audio.DeviceListGet()

Результат:

```
[{«IndexInt»:1, «NameStr»: «»,
    «HostApiInt»: 0, «HostApiStr»: «MME»|»Windows WASAPI»|»Windows WDM-KS»,
    «MaxInputChannelsInt»: 0, «MaxOutputChannelsInt»: 0, «DefaultSampleRateFloat»: 44100.0

},...]
```

Тип результата:
list

pyOpenRPA.Robot.Audio.DeviceMicrophoneIndex() [исходный код]

L-,W+: Выполнить поиск устройства, с помощью которого можно будет выполнить захват с микрофона.

pyOpenRPA.Robot.Audio.DeviceSystemSoundIndex() [исходный код]

L-,W+: Выполнить поиск устройства, с помощью которого можно будет выполнить захват аудио, которое поступает из приложений. Например: аудиоконференции Zoom, whatsapp, telegram и т.д.

class pyOpenRPA.Robot.Audio.Recorder(*inDeviceInt=None*) [исходный код]

Methods:

`CaptureChunk`([inExtra, inForceChunkBool, ...])	L-,W+: Зафиксировать захват аудио в виде промежуточного
`CaptureStart`([inFolderPathStr, ...])	L-,W+: Начать запись звука
`CaptureStop`([inWaitStream, inExtra])	L-,W+: Остановить захват аудио
`CaptureWait`([inWaitCallbackChunkBool, ...])	L-,W+: Ожидать окончания захвата аудио.
`FileInfoGet`([inFileNameStr])	L-,W+: Вернуть информацию по аудиофайлу inFileNameStr.
`FileLastGet`()	L-,W+: Вернуть наименование последнего сохраненного ауд
`FileListGet`()	L-,W+: Вернуть список сохраненных аудиофайлов (наименов
`StatusGet`()	L-,W+: Вернуть статус записи звука

CaptureChunk(*inExtra=None*, *inForceChunkBool=True*, *inShiftSecFloat=0.0*) [исходный код]

L-,W+: Зафиксировать захват аудио в виде промежуточного файла вида: <имя файла>_00000.mp3

Параметры:

- **inExtra** (*any*, *опционально*) – Дополнительный контент, необходимый для идентификации файла. В дальнейшем получить структуру можно с помощью функции FileInfoGet()[„Extra“], по умолчанию None
- **inForceChunkBool** (*bool*, *опционально*) – True - вне зависимости от текущего режима перейти на режим сохранения по частям, по умолчанию True
- **inShiftSecFloat** (*float*) – Последние секунды, которые не записывать в промежуточный аудиофайл. Они будут началом следующего аудио отрывка, по умолчанию 0.0

Результат:
Наименование сохраненного аудиофайла

Тип результата:

str

CaptureStart(*inFolderPathStr=''*, *inFileNameStr='out'*, *inFileFormatStr='mp3'*, *inDurationSecFloat=None*, *inChunkSecFloat=300.0*, *inCallbackChunkDef=None*, *inCallbackStopDef=None*) [исходный код]

L-,W+: Начать запись звука

```
def CallbackChunk(lRec, lFilenameStr):
    pass # КОД ОБРАБОТКИ ПОСЛЕ СОХРАНЕНИЯ ЧАСТИ

from pyOpenRPA.Robot import Audio
lRec = Audio.Recorder()
lRec.CaptureStart(inFileNameStr = "out", inFileFormatStr = "mp3", inDurationSecFloat = None, inChu
lRec.CaptureStop()
```

Параметры:

- **inFolderPathStr** (*str*, *опционально*) – Путь к папке, в которую сохранять аудиофайлы захвата , по умолчанию «»
- **inFileNameStr** (*str*, *опционально*) – Наименование файла без расширения, по умолчанию «out»
- **inFileFormatStr** (*str*, *опционально*) – Наименование формата, в который будет происходить сохранение («mp3» или «wav» или «raw» или «aif»), по умолчанию «mp3»
- **inDurationSecFloat** (*float*, *опционально*) – Длительность захвата аудио, по умолчанию None (пока не поступит команда CaptureStop())
- **inChunkSecFloat** (*float*, *опционально*) – Максимальная длина части аудиофайла, по умолчанию 300.0
- **inCallbackChunkDef** (*def*, *опционально*) – Функция, которая будет инициирована в случае выполнения Chunk сохранения (сохранение части). Callback функция должна принимать 2 аргумента: экземпляр класса Recorder и наименование сохраненного файла. Внимание! Функция запускается асинхронно!
- **inCallbackStopDef** (*def*, *опционально*) – Функция, которая будет инициирована в случае окончания записи. Callback функция должна принимать 2 аргумента: экземпляр класса Recorder и наименование сохраненного файла. Внимание! Функция запускается асинхронно!

CaptureStop(*inWaitStream=True*, *inExtra=None*) [исходный код]

L-,W+: Остановить захват аудио

Параметры:

- **inWaitStream** (*bool*, *опционально*) – True - выполнить ожидание окончания потока захвата перед окончанием, по умолчанию True
- **inExtra** (*any*, *опционально*) – Дополнительный контент, необходимый для идентификации файла. В дальнейшем получить структуру можно с помощью функции FileInfoGet()[„Extra“], по умолчанию None

CaptureWait(*inWaitCallbackChunkBool=True*, *inWaitCallbackStopBool=True*) [исходный код]

L-,W+: Ожидать окончания захвата аудио. Дополнительно настраивается ожидание окончания всех callback функций.

Параметры:

- **inWaitCallbackChunkBool** (*bool*, *опционально*) – True - ожидать выполнение всех асинхронных callback по сохранению части аудиофайла, по умолчанию True
- **inWaitCallbackStopBool** (*bool*, *опционально*) – True - ожидать выполнение всех асинхронных callback по завершению записи, по умолчанию True

FileInfoGet(*inFileNameStr=None*) [исходный код]

L-,W+: Вернуть информацию по аудиофайлу inFileNameStr. Если inFileNameStr == None -> Функция вернет информацию по последнему записанному файлу

Параметры:
inFileNameStr (*str, опционально*) – Наименование аудиофайла с указанием расширения, по умолчанию None (взять последний записанный файл)

Результат:
{StartSecFloat:, EndSecFloat:, Extra:, PathStr:, } или None

Тип результата:
dict

FileLastGet() [исходный код]

L-,W+: Вернуть наименование последнего сохраненного аудиофайла

Результат:
[«out_00000.mp3», «out_00001.mp3», ...]

Тип результата:
list

FileListGet() [исходный код]

L-,W+: Вернуть список сохраненных аудиофайлов (наименования)

Результат:
[«out_00000.mp3», «out_00001.mp3», ...]

Тип результата:
list

StatusGet() [исходный код]

L-,W+: Вернуть статус записи звука

Результат:
«0_READY» или «1_RECORDING»

Тип результата:
str

Быстрая навигация

- Сообщество pyOpenRPA (telegram)
- Сообщество pyOpenRPA (tenchat)
- Сообщество pyOpenRPA (вконтакте)
- Презентация pyOpenRPA
- Портал pyOpenRPA
- Репозиторий pyOpenRPA

.. v1.4.0 replace:: v1.4.0

Предыдущая Следующая

9. Как использовать?

Модуль РОБОТ - это ключевое звено, которое отвечает за продуктивную роботизацию процесса. Данный модуль не имеет графический или консольный интерфейс - он подключается в качестве библиотеки в проект робота, что позволяет выполнять операции максимально быстро. А также позволяет с легкостью интегрировать робота в другие проекты.

Быстрый запуск (Quickstart)

Платформа pyOpenRPA содержит инструменты быстрого прототипирования роботов. Вы можете провести серию экспериментов роботизации без развертывания полноценной инфраструктуры робота - в рамках одного уже преднастроенного файла.

Для быстрого запуска робота необходимо:

- Открыть Jupyter-notebooks: GITToolsJupyter-notebooksstart.cmd (для Windows) или GITToolsJupyter-notebooksstart.sh (для Linux). Откроется окно консоли, в которой будет отображен адрес для входа на веб страницу Jupyter-notebooks.
- В web окне Jupyter-notebooks открыть QuickstartRobot.ipynb

Быстрая инфраструктура для прототипирования робота готова!

В файле Robot.ipynb содержится вся необходимая информация, которая позволит решить любую поставленную задачу.

Как запустить скрипт робота?

Запустить скрипт робота можно 2-мя способами:

- Скрипт Python (файл .py)
- Скрипт в Студии pyOpenRPA
- Скрипт в Jupyter (см. раздел «Быстрый запуск»)

Скрипт Python (файл .py)

Чтобы начать использовать модуль робота достаточно выполнить в файле скрипта соответсвующие команды импорта:

```
import sys
sys.path.append('../../')
from pyOpenRPA.Robot import UIDesktop # Взаимодействие с UI объектами приложений
from pyOpenRPA.Robot import UIWeb # Взаимодействие с UI объектами веб приложений
from pyOpenRPA.Robot import Keyboard # Взаимодействие с клавиатурой
from pyOpenRPA.Robot import Clipboard # Взаимодействие с буфером обмена
from pyOpenRPA.Robot import Mouse # Взаимодействие с мышью
from pyOpenRPA.Robot import Image # Взаимодействие с графической сессией ОС
```

Описание каждого из этих модулей представлены в разделе «МОДУЛЬ РОБОТ»

Execute python script

pyOpenRPA - это максимально инкапсулированная платформа программной роботизаии RPA. Все необходимые зависимости находятся внутри нее, что позволяет копировать робота между ЭВМ максимально просто.

Вы можете запустить скрипт робота RPA следующими способами:

- Запустить из интерпретатора Python x32 (ResourcesWPy32-3720python-3.7.2python.exe)
- Запустить из интерпретатора Python x64 (ResourcesWPy64-3720python-3.7.2.amd64python.exe)
- Запустить из под .cmd файла

Запустить из интерпретатора Python x32

Для запуска скрипта из интерпретатора Python x32 необходимо открыть командную строку (cmd), и выполнить следующие команды:

```
cd "Resources\WPy32-3720\python-3.7.2" # Установить рабочую директорию там, где находится интерпретатор P
python.exe "path to your python script.py" # Запустить интерпретатор Python с файлом скрипта робота "path
```

Запустить из интерпретатора Python x64

Для запуска скрипта из интерпретатора Python x64 необходимо открыть командную строку (cmd), и выполнить следующие команды:

```
cd "Resources\WPy32-3720\python-3.7.2.amd64" # Установить рабочую директорию там, где находится интерпрет
python.exe "path to your python script.py" # Запустить интерпретатор Python с файлом скрипта робота "path
```

Запустить из под .cmd файла

Упростить процесс запуска и свести инициализацию робота к одному нажатию можно с помощью средстве command shell и .cmd файла.

Для этого достаточно выбрать рабочую директорию робота, там создать текстовый .cmd файл, и прописать в нем следующий код:

```
cd %~dp0 # Установить рабочую директорую там, где находится этот .cmd файл
copy /Y ..\Resources\WPy32-3720\python-3.7.2\python.exe ..\Resources\WPy32-3720\python-3.7.2\OpenRPAOrche
.\..\Resources\WPy32-3720\python-3.7.2\OpenRPAOrchestrator.exe orchestratorMain.py # Выполнить инициализа
pause >nul # Не закрывать окно консоли после завершения работы скрипта робота
```

Быстрая навигация

- Сообщество pyOpenRPA (telegram)
- Сообщество pyOpenRPA (tenchat)
- Сообщество pyOpenRPA (вконтакте)
- Презентация pyOpenRPA
- Портал pyOpenRPA
- Репозиторий pyOpenRPA

.. v1.4.0 replace:: v1.4.0

Предыдущая Следующая

1. Описание

Общее

Модуль студии обеспечивает всю необходимую функциональность взаимодействия с UI объектами различных приложений, запущенных в сессии.

В разделе 2. Как использовать? вы найдете практическое руководство по работе с пользовательским интерфейсом студии, которая отрывается в веб браузере.

Дорогие коллеги!

Мы знаем, что с pyOpenRPA вы сможете существенно улучшить качество вашего бизнеса. Платформа роботизации pyOpenRPA - это разработка, которая дает возможность делать виртуальных сотрудников (программных роботов RPA) выгодными, начиная от эффекта всего в **10 тыс. руб.** И управлять ими будете только Вы!

Если у вас останутся вопросы, то вы всегда можете обратиться в центр поддержки клиентов pyOpenRPA. Контакты: 2. Лицензия & Контакты

!ВНИМАНИЕ! КОМПОНЕНТ СТУДИИ ДОСТУПЕН ТОЛЬКО В WINDOWS. ДЛЯ LINUX КОНФИГУРАЦИЙ НЕ ТРЕБУЕТСЯ

pyOpenRPA - роботы помогут!

Быстрая навигация

- Сообщество pyOpenRPA (telegram)
- Сообщество pyOpenRPA (tenchat)
- Сообщество pyOpenRPA (вконтакте)
- Презентация pyOpenRPA
- Портал pyOpenRPA
- Репозиторий pyOpenRPA

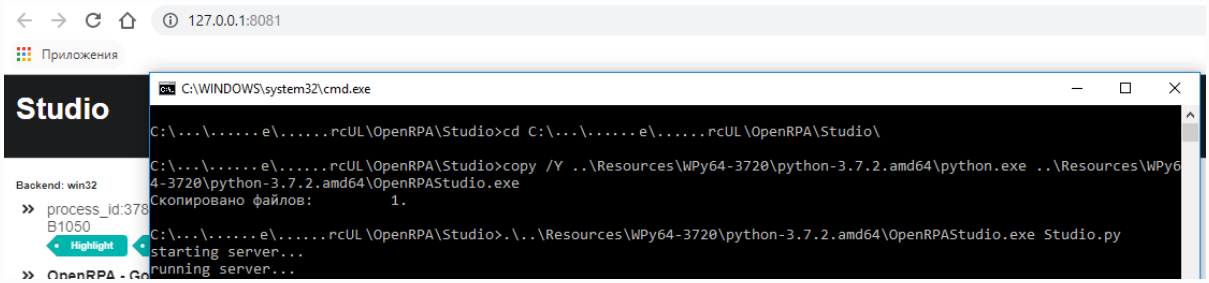
.. v1.4.0 replace:: v1.4.0

2. Как использовать?

Общее

Как запустить?

- Запустить файл Studiostart.cmd
- Ожидать текст в окне консоли: «running server». Браузер, установленный по умолчанию откроется автоматически
- **!ВНИМАНИЕ!** Студия поддерживает все версии браузеров, кроме Internet Explorer.



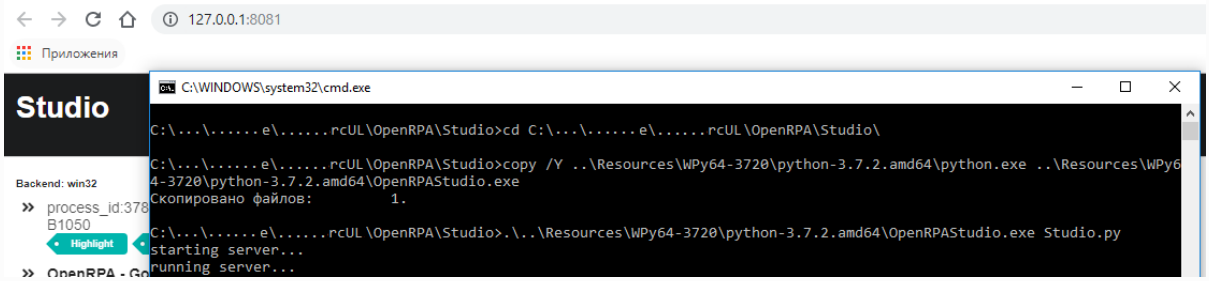
Описание UI студии

Интерфейс (UI) студии состоит из следующих компонентов:

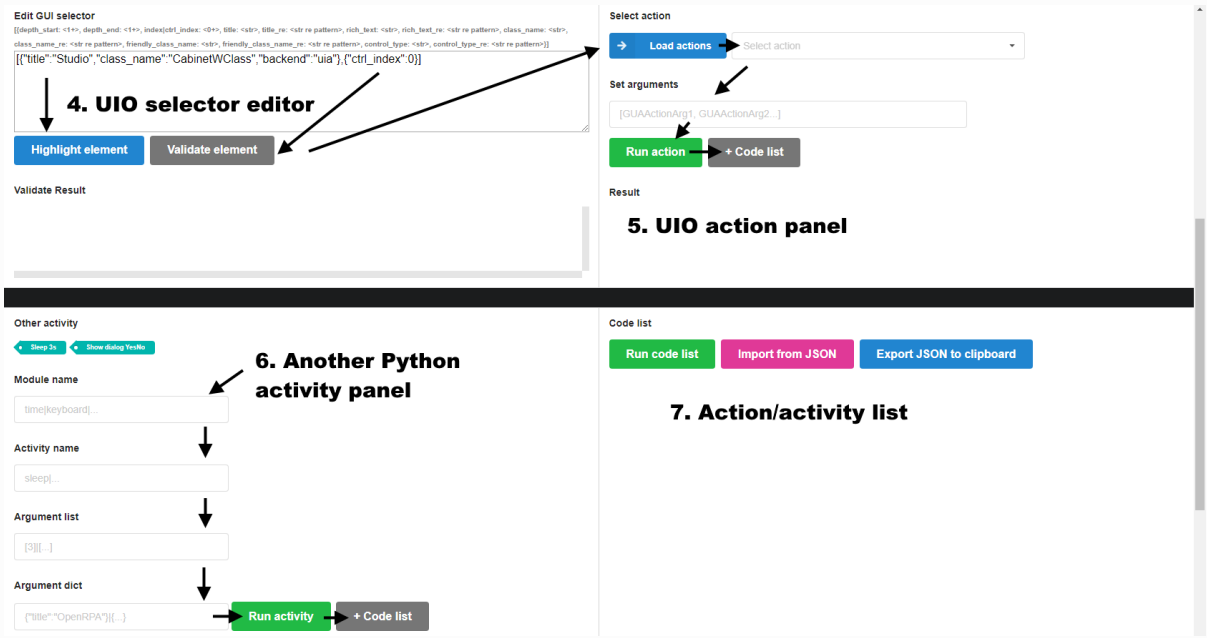
1. Обозреватель UI дерева
2. Обозреватель иерархии выбранного UI объекта
3. Обозреватель свойств выбранного уровня UI объекта
4. Редактор UIO селектора
5. Панель активностей над UIO объектом
6. Панель других Python активностей
7. Список активностей

Ниже представлены скриншоты студии

Скриншот 1



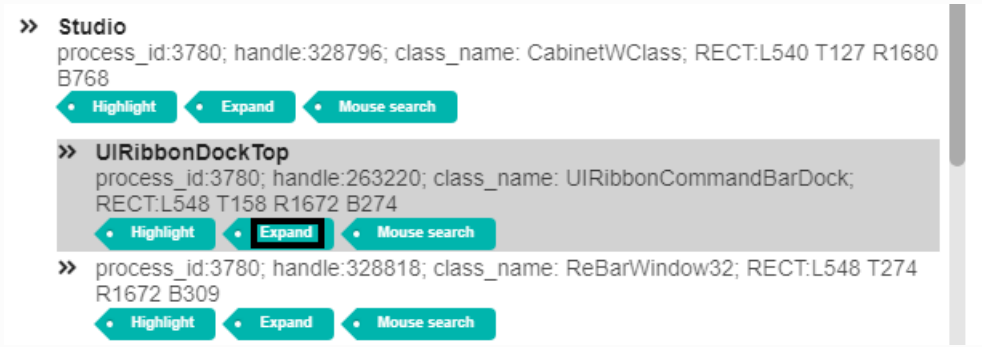
Скриншот 2



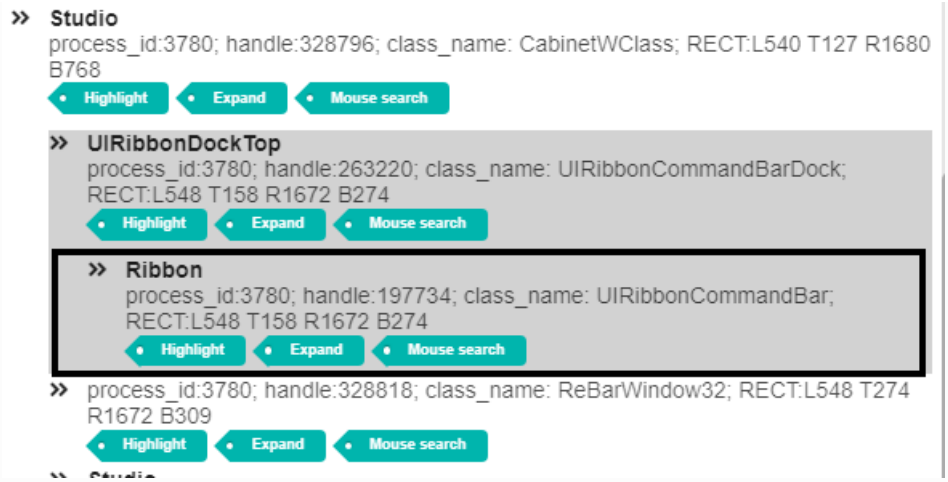
Извлечь UI дерево

Чтобы извлечь дерево пользовательского интерфейса выполните следующие действия: в `UI Tree viewer` выбрать интересующий UI объект и нажать кнопку `Expand`.

Действие: Нажать по кнопке «Expand»



Итог

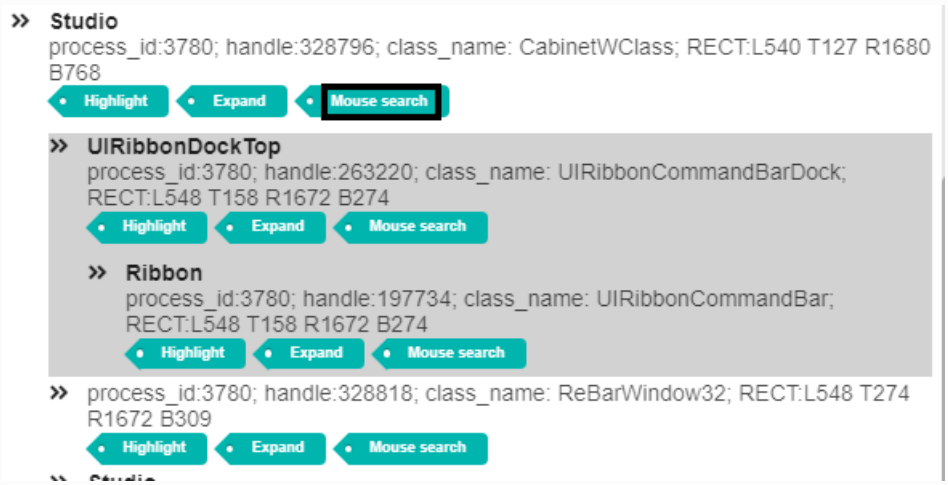


Поиск UI объекта по наведению мыши

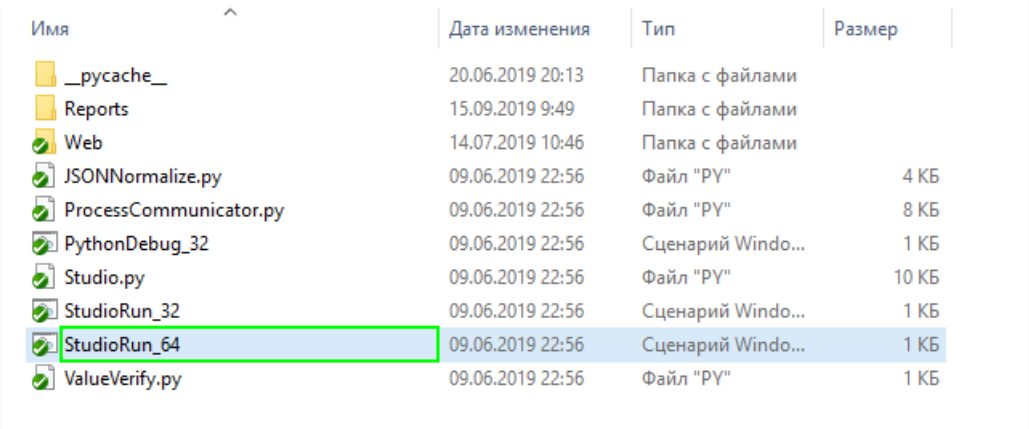
Чтобы выполнить поиск UI объекта пользовательского интерфейса, необходимо в `UI tree viewer` выбрать родительский UI объект, в котором вы хотите выполнить поиск, и нажать кнопку `Mouse search`.

Будет активирован режим поиск UI объекта по курсору мыши. Наведите курсор мыши на интересующий вас объект пользовательского интерфейса и дождитесь, когда студия выделит объект пользовательского интерфейса. После выделения цветов удерживайте клавишу «Ctrl» и подождите 3 секунды. Интересующий UI объект будет показан в `UI tree viewer`.

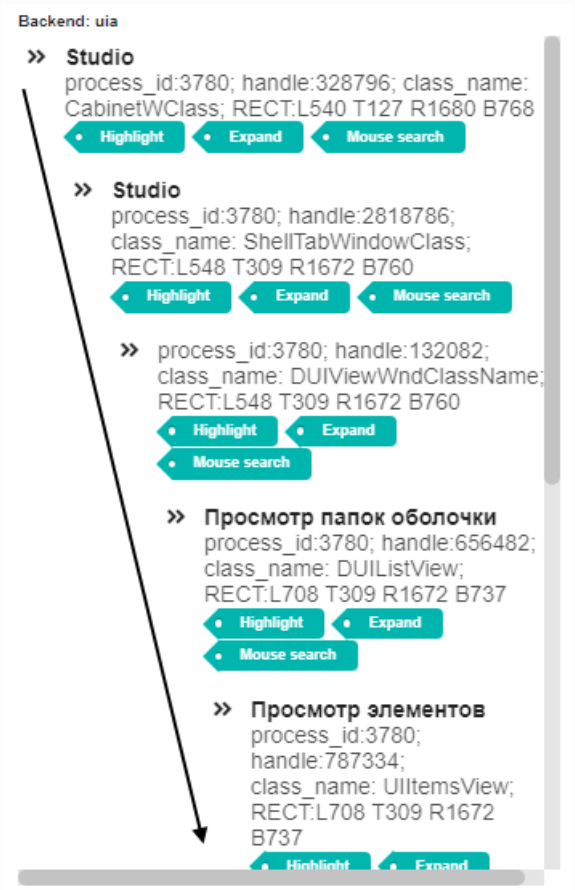
Действие: Нажать кнопку «Mouse search»



Действие: Навести курсор мыши на UI объект, который интересует и зажать клавишу «Ctrl» на 3 секунды



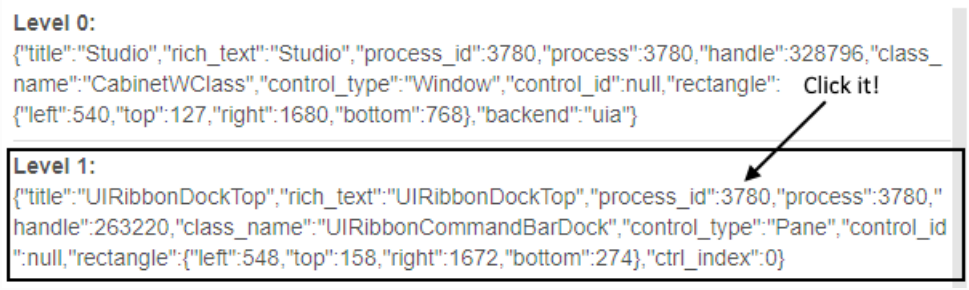
Итог: Интересующий UI объект будет отображен в `UI tree viewer`



Извлечь свойства UI объекта

Чтобы извлечь свойства UI объекта, необходимо в `Selected UI object hierarchy list` выбрать интересующий UI объект и щелкнуть по нему. Список свойств UI объекта будет отображен в `Selected UI object property list`

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



Итог: Свойства UI объекта будут отображены в

`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: ""

Быстрая навигация

- Сообщество pyOpenRPA (telegram)
- Сообщество pyOpenRPA (tenchat)
- Сообщество pyOpenRPA (вконтакте)
- Презентация pyOpenRPA
- Портал pyOpenRPA
- Репозиторий pyOpenRPA

.. v1.4.0 replace:: v1.4.0

Предыдущая

Следующая

1. Описание

Общее

Модуль оркестратор - это координирующее звено, которое обеспечивает бесперебойную работа массива роботов. Этот массив может состоять как из одного так и из тысячи роботов RPA.

Основные возможности

- Запуск / пауза / безопасная остановка / принудительная остановка робота
- Интеллектуальное расписание
- Просмотр состояния графических сессий роботов через панель управления
- Удаленное администрирование сессий оркестратора и робота
- Среда отладки функциональности через панель управления оркестратора
- Консолидированное хранилище логов, доступное для просмотра через панель управления
- Ролевая модель разграничения доступа
- Функциональность очередей для координации роботов

В качестве основы для web сервера используется один из самых прогрессивных и производительных фреймворков от FastAPI.

Пример использования FastAPI см. в GIT/Orchestrator/config.py

Концепция единого глобального словаря настроек (GSettings)

pyOpenRPA - это сложное решение, которое направлено на упрощение жизни пользователей и разработчиков роботов.

Для того, чтобы предлагать рынку гибкое, адаптивное и надежное решение, одним из архитектурных решений был выбран подхъод хранения !ВСЕЙ! конфигурационной информации в едином словаре, который мы называем GSettings.

GSettings - это многоуровневая и иерархичная структура, которая позволяет произвести широкую кастомизацию под свои нужды, и в то же время быть открытой к внедрению новых возможностей.

Ознакомиться со структурой GSettings можно по ссылке: 3. Настройки GSettings (шаблон)

Мы не рекомендуем вносить изменения напрямую в GSettings, хоть мы и оставляем такую возможность. Для корректировки функциональности Вы можете воспользоваться соответствующей функцией в модуле Оркестратора (см. здесь: 2. Функции)

Используя специальные функции модуля Оркестратора вы существенно увеличиваете шансы бесшовного перехода на новые версии pyOpenRPA, если вам это потребуется.

Архитектура

Оркестратор состоит из следующих основных потоков:

- Процессорная очередь активностей (ActivityItem) (Processor)
- Функциональность асинхронного исполнения активностей (ActivityItem) (Processor)
- Поток интеллектуального расписания (main)
- Поток контроля активности RDP сессий
- Поток сбора мусорных данных
- Поток контроля графической сессии на учетной записи, где работает Оркестратор
- Поток веб-сервера Оркестратора

Ознакомиться с возможностями и функциями оркестратора можно по ссылке: 2. Функции

Дорогие коллеги!

Мы знаем, что с pyOpenRPA вы сможете существенно улучшить качество вашего бизнеса. Платформа роботизации pyOpenRPA - это разработка, которая дает возможность делать виртуальных сотрудников (программных роботов RPA) выгодными, начиная от эффекта всего в **10 тыс. руб.** И управлять ими будете только Вы!

Если у вас останутся вопросы, то вы всегда можете обратиться в центр поддержки клиентов pyOpenRPA. Контакты: 2. Лицензия & Контакты

pyOpenRPA - роботы помогут!

Быстрая навигация

- Сообщество pyOpenRPA (telegram)
- Сообщество pyOpenRPA (tenchat)
- Сообщество pyOpenRPA (вконтакте)
- Презентация pyOpenRPA
- Портал pyOpenRPA
- Репозиторий pyOpenRPA

.. v1.4.0 replace:: v1.4.0

Предыдущая Следующая

2. Функции

Общее

Раздел содержит всю необходимую информацию о функциях pyOpenRPA.Orchestrator

При необходимости вы всегда можете обратиться в центр поддержки клиентов pyOpenRPA.
Контакты: 2. Лицензия & Контакты

Что такое активность (ActivityItem)?

Архитектура pyOpenRPA позволяет обмениваться сообщениями о выполнении функций через механизм активностей (ActivityItem).

На стороне Агента и Оркестратора реализована процессорная очередь, которая последовательно выполняет поставленные активности. Результат этих активностей сообщается инициатору (см. функции группы Agent... в Оркестраторе)

Функции

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

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

Группа функций Agent...

Взаимодействие между Оркестратором и Агентом, который развертнут на других графических сессиях, где будет происходить запуск робота.

Группа функций GSettings...

Вспомогательные функции для работы с глобальным словарем настроек Оркестратора

Группа функций Storage...

Функции для взаимодействия со специальным хранилищем переменных, предназначенного для хранения информации от роботов.

!ВНИМАНИЕ! Данное хранилище сохраняется при перезагрузке Оркестратора из панели управления.

Группа функций OS...

Функции взаимодействия с командной строкой на сессии, где запущен Оркестратор.

Группа функций Process...

Запуск / остановка процессов на сессии Оркестратора.

Группа функций Processor...

Функции взаимодействия с процессорной очередью. Если требуется выполнить синхронизацию нескольких разных задач, то можно их отправлять в процессорную очередь.

Группа функций Python...

Функции взаимодействия с Python модулями.

Группа функций RDPSession...

Запуск, отключение, перезапуск, отправка CMD команд, раскрыть на весь экран на RDP сессию

Группа функций Web...

Управление веб-сервером Оркестратора.

Группа функций UAC...

Управление ролевой моделью доступа пользователей к панели управления Оркестратора. Актуально для подключения бизнес-пользователей.

Группа функций Scheduler...

Установка расписания на различные активности.

Описание каждой функции начинается с обозначения L+,W+, что означает, что функция поддерживается в ОС Linux (L) и поддерживается в Windows (W)

Functions:

`ActivityItemCreate` (inDef[, inArgList, ...])	L+,W+: Создать Активность (ActivityItem).
`ActivityItemDefAliasCreate` (inDef[, ...])	L+,W+: Создать синоним (текстовый ключ) для инициации
`ActivityItemDefAliasModulesLoad` ()	L+,W+: Загрузить все функции из импортированных модуле
`ActivityItemDefAliasUpdate` (inDef, inAliasStr)	L+,W+: Обновить синоним (текстовый ключ) для инициаци
`ActivityItemHelperDefAutofill` (inDef)	L+,W+: Анализ аргументов функции по синониму (текстово
`ActivityItemHelperDefList` ([inDefQueryStr])	L+,W+: Получить список синонимов (текстовых ключей), до
`AgentActivityItemAdd` (inHostNameStr, ...[, ...])	L+,W+: Добавить активность в словарь активностей выбран
`AgentActivityItemExists` (inHostNameStr, ...)	L+,W+: Выполнить проверку, что активность (ActivityItem) б
`AgentActivityItemReturnExists` (inGUIDStr[, ...])	L+,W+: Выполнить проверку, что активность (ActivityItem) б
`AgentActivityItemReturnGet` (inGUIDStr[, ...])	L+,W+: Ожидает появления результата по активности (Activ
`AgentOSCMD` (inHostNameStr, inUserStr, inCMDStr)	L+,W+: Отправка команды командной строки на сессию, гд
`AgentOSFileBinaryDataBase64StrAppend` (...[, ...])	L+,W+: Добавить бинарную информацию в существующий
`AgentOSFileBinaryDataBase64StrCreate` (...[, ...])	L+,W+: Создать бинарный файл, который будет расположе

`AgentOSFileBinaryDataBase64StrReceive` (...[, ...])	L+,W+: Выполнить чтение бинарного файла и получить сод
`AgentOSFileBinaryDataBytesCreate` (...[, ...])	L+,W+: Создать бинарный файл, который будет расположе
`AgentOSFileBinaryDataReceive` (inHostNameStr, ...)	L+,W+: Чтение бинарного файла на стороне Агента по адре
`AgentOSFileSend` (inHostNameStr, inUserStr, ...)	L+,W+: Отправить файл по адресу inOrchestratorFilePathStr
`AgentOSFileTextDataStrCreate` (inHostNameStr, ...)	L+,W+: Создать текстовый файл, который будет расположе
`AgentOSFileTextDataStrReceive` (inHostNameStr, ...)	L+,W+: Чтение текстового файла на стороне Агента по адре
`AgentOSLogoff` (inHostNameStr, inUserStr)	L+,W+: Выполнить операцию logoff на стороне пользовател
`AgentProcessWOExeUpperUserListGet` (...[, ...])	L-,W+: Получить список процессов, которые выполняется н
`GSettingsGet` ([inGSettings])	L+,W+: Вернуть глобальный словарь настроек Оркестратор
`GSettingsKeyListValueAppend` (inValue[, ...])	L+,W+: Применить операцию .append к обьекту, расположе
`GSettingsKeyListValueGet` ([inKeyList, ...])	L+,W+: Получить значение из глобального словаря настрое
`GSettingsKeyListValueOperatorPlus` (inValue[, ...])	L+,W+: Применить оператор сложения (+) к обьекту, распол
`GSettingsKeyListValueSet` (inValue[, ...])	L+,W+: Установить значение из глобального словаря настр
`OSCMD` (inCMDStr[, inRunAsyncBool, inLogger])	L-,W+: Отправить команду на выполнение на сессию, где в
`OSCredentialsVerify` (inUserStr, inPasswordStr)	L+,W+: Выполнить верификацию доменного (локального) п
`OSLogoff` ()	L+,W+: Выполнить отключение сессии, на которой выполня
`OSRemotePCRestart` (inHostStr[, inForceBool, ...])	L-,W+: Отправить сигнал на удаленную перезагрузку опера
`OSRestart` ([inForceBool, inLogger])	L+,W+: Отправить сигнал на перезагрузку операционной си
`Orchestrator` ([inGSettings, ...])	L+,W+: Инициализация ядра Оркестратора (всех потоков)
`OrchestratorInitWait` ()	L+,W+: Ожидать инициализацию ядра Оркестратора
`OrchestratorIsAdmin` ()	L+,W+: Проверить, запущен ли Оркестратор с правами адм
`OrchestratorIsCredentialsAsk` ()	L+,W+: Проверить, активирована ли авторизация при перех
`OrchestratorIsInited` ()	L+,W+: Проверить, было ли проинициализировано ядро Ор
`OrchestratorLoggerGet` ()	L+,W+: Получить логгер Оркестратора
`OrchestratorPySearchInit` (inGlobPatternStr[, ...])	L+,W+: Выполнить поиск и инициализацию пользовательск
`OrchestratorRerunAsAdmin` ()	L-,W+: Перезапустить Оркестратор с правами локального а
`OrchestratorRestart` ([inGSettings])	L+,W+: Перезапуск Оркестратора с сохранением информац
`OrchestratorScheduleGet` ()	L+,W+: Базовый объект расписания, который можно исполи
`OrchestratorSessionRestore` ([inGSettings])	L+,W+: Восстановить состояние Оркестратора, если ранее
`OrchestratorSessionSave` ([inGSettings])	L+,W+: Сохранить состояние Оркестратора (для дальнейше
`OrchestratorThreadStart` (inDef, *inArgList, ...)	L+,W+: Запустить функцию в отдельном потоке.
`ProcessDefIntervalCall` (inDef, inIntervalSecFloat)	L+,W+: Периодический вызов функции Python.
`ProcessIsStarted` (inProcessNameWOExeStr)	L-,W+: Проверить, запущен ли процесс, который в наимено
`ProcessListGet` ([inProcessNameWOExeList])	L-,W+: Вернуть список процессов, запущенных на ОС, где р
`ProcessStart` (inPathStr, inArgList[, ...])	L-,W+: Запуск процесса на сессии Оркестратора, если на О
`ProcessStop` (inProcessNameWOExeStr, ...[, ...])	L-,W+: Остановить процесс на ОС, где работает Оркестрат

`ProcessorActivityItemAppend` ([inGSettings, ...])	L+,W+: Добавить активность (ActivityItem) в процессорную
`ProcessorActivityItemCreate` (inDef[, ...])	L+,W+: Создать Активность (ActivityItem).
`ProcessorAliasDefCreate` (inDef[, inAliasStr, ...])	L+,W+: Создать синоним (текстовый ключ) для инициации
`ProcessorAliasDefUpdate` (inDef, inAliasStr[, ...])	L+,W+: Обновить синоним (текстовый ключ) для инициаци
`PythonStart` (inModulePathStr, inDefNameStr[, ...])	L+,W+: Импорт модуля и выполнение функции в процессе
`RDPSessionCMDRun` (inRDPSessionKeyStr, inCMDStr)	L-,W+: Отправить CMD команду на удаленную сесиию чере
`RDPSessionConnect` (inRDPSessionKeyStr[, ...])	L-,W+: Выполнить подключение к RDP и следить за стабил
`RDPSessionDisconnect` (inRDPSessionKeyStr[, ...])	L-,W+: Выполнить отключение RDP сессии и прекратить мо
`RDPSessionFileStoredRecieve` (...[, inGSettings])	L-,W+: Получение файла со стороны RDP сессии на сторон
`RDPSessionFileStoredSend` (inRDPSessionKeyStr, ...)	L-,W+: Отправка файла со стороны Оркестратора на сторо
`RDPSessionLogoff` (inRDPSessionKeyStr[, ...])	L-,W+: Выполнить отключение (logoff) на RDP сессии и пре
`RDPSessionMonitorStop` (inRDPSessionKeyStr[, ...])	L-,W+: Прекратить мониторить активность RDP соединени
`RDPSessionProcessStartIfNotRunning` (...[, ...])	L-,W+: Выполнить запуск процесса на RDP сессии через гра
`RDPSessionProcessStop` (inRDPSessionKeyStr, ...)	L-,W+: Отправка CMD команды в RDP окне на остановку пр
`RDPSessionReconnect` (inRDPSessionKeyStr[, ...])	L-,W+: Выполнить переподключение RDP сессии и продол
`RDPTemplateCreate` (inLoginStr, inPasswordStr)	L-,W+: Создать шаблон подключения RDP (dict).
`SchedulerActivityTimeAddWeekly` ([...])	L+,W+: Добавить активность по расписанию.
`StorageRobotExists` (inRobotNameStr)	L+,W+: Проверить, существует ли ключ inRobotNameStr в х
`StorageRobotGet` (inRobotNameStr)	L+,W+: Получить содержимое по ключу робота inRobotNam
`UACKeyListCheck` (inRequest, inRoleKeyList)	L+,W+: [ПРЕКРАЩЕНИЕ ПОДДЕРЖКИ В 1.3.2, см.
`UACSuperTokenUpdate` (inSuperTokenStr[, ...])	L+,W+: Добавить супертокен (полный доступ).
`UACUpdate` (inADLoginStr[, inADStr, ...])	L+,W+: Дообогащение словаря доступа UAC пользователя
`UACUserDictGet` (inRequest)	L+,W+: [ПРЕКРАЩЕНИЕ ПОДДЕРЖКИ В 1.3.2, см.
`WebAppGet` ()	L+,W+: Вернуть экземпляр веб сервера fastapi.FastAPI (app).
`WebAuditMessageCreate` ([inAuthTokenStr, ...])	L+,W+: [ИЗМЕНЕНИЕ В 1.3.1] Создание сообщения ИТ ауди
`WebAuthDefGet` ()	Вернуть функцию авторизации пользователя.
`WebCPUpdate` (inCPKeyStr[, inHTMLRenderDef, ...])	L+,W+: Добавить панель управления робота в Оркестратор
`WebListenCreate` ([inServerKeyStr, ...])	L+,W+: Настроить веб-сервер Оркестратора.
`WebRequestGet` ()	L+,W+: [ПРЕКРАЩЕНИЕ ПОДДЕРЖКИ В 1.3.2] Вернуть экз
`WebRequestHostGet` (inRequest)	L+,W+: Получить наименование хоста, с которого поступил
`WebRequestParseBodyBytes` ([inRequest])	L+,W+: [ПРЕКРАЩЕНИЕ ПОДДЕРЖКИ В 1.3.1, см.
`WebRequestParseBodyJSON` ([inRequest])	L+,W+: [ПРЕКРАЩЕНИЕ ПОДДЕРЖКИ В 1.3.1, см.
`WebRequestParseBodyStr` ([inRequest])	L+,W+: [ПРЕКРАЩЕНИЕ ПОДДЕРЖКИ В 1.3.1, см.
`WebRequestParseFile` ([inRequest])	L+,W+: [ПРЕКРАЩЕНИЕ ПОДДЕРЖКИ В 1.3.1, см.
`WebRequestParsePath` ([inRequest])	L+,W+: [ПРЕКРАЩЕНИЕ ПОДДЕРЖКИ В 1.3.1, см.
`WebRequestResponseSend` (inResponeStr[, ...])	L+,W+: [ПРЕКРАЩЕНИЕ ПОДДЕРЖКИ В 1.3.1, см.
`WebURLConnectDef` (inMethodStr, inURLStr, ...)	L+,W+: Подключить функцию Python к URL.

`WebURLConnectFile` (inMethodStr, inURLStr, ...)	L+,W+: Подключить файл к URL.
`WebURLConnectFolder` (inMethodStr, inURLStr, ...)	L+,W+: Подключить папку к URL.
`WebURLIndexChange` ([inURLIndexStr])	L+,W+: Изменить адрес главной страницы Оркестратора.
`WebUserDomainGet` ([inAuthTokenStr])	L+,W+: Получить домен авторизованного пользователя.
`WebUserInfoGet` ([inAuthTokenStr])	L+,W+: Информация о пользователе, который отправил HT
`WebUserIsSuperToken` ([inAuthTokenStr])	L+,W+: [ИЗМЕНЕНИЕ В 1.3.1] Проверить, авторизован ли H
`WebUserLoginGet` ([inAuthTokenStr])	L+,W+: Получить логин авторизованного пользователя.
`WebUserUACCheck` ([inAuthTokenStr, inKeyList])	L+,W+: Проверить UAC доступ списка ключей для пользова
`WebUserUACHierarchyGet` ([inAuthTokenStr])	L+,W+: [ИЗМЕНЕНИЕ В 1.3.1] Вернуть словарь доступа UA(

pyOpenRPA.Orchestrator.__Orchestrator__.ActivityItemCreate(*inDef, inArgList=None, inArgDict=None, inArgGSettingsStr=None, inArgLoggerStr=None, inGUIDStr=None, inThreadBool=False*) [исходный код]

L+,W+: Создать Активность (ActivityItem). Активность можно использовать в ProcessorActivityItemAppend или в Processor.ActivityListExecute или в функциях работы с Агентами.

```
# ПРИМЕР
from pyOpenRPA import Orchestrator

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

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

Параметры:

- **inDef** – Функция Python или синоним (текстовый ключ)

- **inArgList** – Список (list) неименованных аргументов к функции
- **inArgDict** – Словарь (dict) именованных аргументов к функции
- **inArgGSettingsStr** – Текстовое наименование аргумента GSettings (если требуется передавать)
- **inArgLoggerStr** – Текстовое наименование аргумента logger (если требуется передавать)
- **inGUIDStr** – ГУИД идентификатор активности (ActivityItem). Если ГУИД не указан, то он будет сгенерирован автоматически
- **inThreadBool** – True - выполнить ActivityItem в новом потоке; False - выполнить последовательно в общем потоке процессорной очереди

Результат:

lActivityItemDict= {

«Def»:inDef, # def link or def alias (look gSettings[«Processor»][«AliasDefDict»]) «ArgList»:inArgList, # Args list «ArgDict»:inArgDict, # Args dictionary «ArgGSettings»: inArgGSettingsStr, # Name of GSettings attribute: str (ArgDict) or index (for ArgList) «ArgLogger»: inArgLoggerStr, # Name of GSettings attribute: str (ArgDict) or index (for ArgList) «GUIDStr»: inGUIDStr, «ThreadBool»: inThreadBool

}

pyOpenRPA.Orchestrator.__Orchestrator__.ActivityItemDefAliasCreate(*inDef, inAliasStr=None, inGSettings=None*) [исходный код]

L+,W+: Создать синоним (текстовый ключ) для инициации выполнения функции в том случае, если запрос на выполнения пришел из вне (передача функций невозможна).

```
# ПРИМЕР
from pyOpenRPA import Orchestrator

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

Параметры:

- **inDef** – функция Python
- **inAliasStr** – Строковый ключ (синоним), который можно будет использовать в Активности (ActivityItem)
- **inGSettings** – Глобальный словарь настроек Оркестратора (синглтон)

Результат:

Строковый ключ, который был назначен. Ключ может быть изменен, если входящий текстовый ключ был уже занят.

pyOpenRPA.Orchestrator.__Orchestrator__.ActivityItemDefAliasModulesLoad() [исходный код]

L+,W+: Загрузить все функции из импортированных модулей sys.modules в ActivityItem синонимы - полезно для отладки со стороны панели управления.

pyOpenRPA.Orchestrator.__Orchestrator__.ActivityItemDefAliasUpdate(*inDef, inAliasStr, inGSettings=None*) [исходный код]

L+,W+: Обновить синоним (текстовый ключ) для инициации выполнения функции в том случае, если запрос на выполнения пришел из вне (передача функций невозможна).

```
# USAGE
from pyOpenRPA import Orchestrator

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

Параметры:

- **inDef** – функция Python
- **inAliasStr** – Строковый ключ (синоним), который можно будет использовать в Активности (ActivityItem)
- **inGSettings** – Глобальный словарь настроек Оркестратора (синглтон)

Результат:

Строковый ключ, который был назначен. Ключ будет тем же, если входящий текстовый ключ был уже занят.

pyOpenRPA.Orchestrator.__Orchestrator__.ActivityItemHelperDefAutofill(*inDef*) [исходный код]

L+,W+: Анализ аргументов функции по синониму (текстовому ключу).

Параметры:

inDef – Часть текстового ключ (начало / середина / конец)

Результат:

Преднастроенная структура активности (ActivityItem) {

»Def»: None, «ArgList»: [], «ArgDict»: {}, «ArgGSettingsStr»: None, «ArgLoggerStr»: None

}

pyOpenRPA.Orchestrator.__Orchestrator__.ActivityItemHelperDefList(*inDefQueryStr=None*) [исходный код]

L+,W+: Получить список синонимов (текстовых ключей), доступных для использования в Активностях (ActivityItem).

Параметры:

inDefStr – Часть текстового ключ (начало / середина / конец)

Результат:

Список доступных ключей в формате: [«ActivityItemDefAliasUpdate», «ActivityItemDefAliasCreate», etc...]

pyOpenRPA.Orchestrator.__Orchestrator__.AgentActivityItemAdd(*inHostNameStr, inUserStr, inActivityItemDict, inGSettings=None*) [исходный код]

L+,W+: Добавить активность в словарь активностей выбранного Агента

Параметры:

- **inGSettings** – Глобальный словарь настроек Оркестратора (синглтон)
- **inHostNameStr** – Наименование хоста, на котором запущен Агент. Наименования подключенных агентов доступно для просмотра в панели управления
- **inUserStr** – Наименование пользователя, на графической сессии которого запущен Агент. Наименования подключенных агентов доступно для просмотра в панели управления
- **inActivityItemDict** – Активность (ActivityItem). См. функцию ProcessorActivityitemCreate

Результат:

ГУИД (GUID) строка Активности (ActivityItem). Далее можно ожидать результат этой функции по ГУИД с помощью функции AgentActivityItemReturnGet

---

pyOpenRPA.Orchestrator.__Orchestrator__.AgentActivityItemExists(*inHostNameStr, inUserStr, inGUIDStr, inGSettings=None*) [исходный код]

L+,W+: Выполнить проверку, что активность (ActivityItem) была отправлена на сторону Агента.

Параметры:

- **inGSettings** – Глобальный словарь настроек Оркестратора (синглтон)
- **inGUIDStr** – ГУИД (GUID) активности (ActivityItem)

Результат:

True - Активность присутствует ; False - Активность еще не была отправлена на сторону Агента

---

pyOpenRPA.Orchestrator.__Orchestrator__.AgentActivityItemReturnExists(*inGUIDStr, inGSettings=None*) [исходный код]

L+,W+: Выполнить проверку, что активность (ActivityItem) была выполнена на стороне Агента и результат был получен на стороне Оркестратора.

Параметры:

- **inGSettings** – Глобальный словарь настроек Оркестратора (синглтон)
- **inGUIDStr** – ГУИД (GUID) активности (ActivityItem)

Результат:

True - Активность присутствует; False - Активность еще не была выполнена на стороне Агента

---

pyOpenRPA.Orchestrator.__Orchestrator__.AgentActivityItemReturnGet(*inGUIDStr, inCheckIntervalSecFloat=0.5, inGSettings=None, inTimeoutSecFloat=None*) [исходный код]

L+,W+: Ожидает появления результата по активности (ActivityItem). Возвращает результат выполнения активности.

!ВНИМАНИЕ! Замораживает поток, пока не будет получен результат. !ВНИМАНИЕ! Запускать следует после того как будет инициализировано ядро Оркестратора (см. функцию OrchestratorInitWait), иначе будет инициирована ошибка.

Параметры:

- **inGSettings** – Глобальный словарь настроек Оркестратора (синглтон)
- **inGUIDStr** – ГУИД (GUID) активности (ActivityItem)
- **inCheckIntervalSecFloat** – Интервал в секундах, с какой частотой выполнять проверку результата. По умолчанию 0.5
- **inTimeoutSecFloat** – Время ожидания ответа. Если ответ не поступил, генерация исключения Exception.

Результат:

Результат выполнения активности. !ВНИМАНИЕ! Возвращаются только то результаты, которые могут быть интерпретированы в JSON формате.

---

pyOpenRPA.Orchestrator.__Orchestrator__.AgentOSCMD(*inHostNameStr, inUserStr, inCMDStr, inRunAsyncBool=True, inSendOutputToOrchestratorLogsBool=True, inCMDEncodingStr='cp1251', inGSettings=None, inCaptureBool=True*) [исходный код]

L+,W+: Отправка команды командной строки на сессию, где работает pyOpenRPA.Agent. Результат выполнения команды можно выводить в лог оркестратора.

Параметры:

- **inGSettings** – Глобальный словарь настроек Оркестратора (синглтон)
- **inHostNameStr** – Наименование хоста, на котором запущен Агент. Наименования подключенных агентов доступно для просмотра в панели управления
- **inUserStr** – Наименование пользователя, на графической сессии которого запущен Агент. Наименования подключенных агентов доступно для просмотра в панели управления
- **inCMDStr** – Команда для исполнения на стороне сессии Агента
- **inRunAsyncBool** – True - Агент не ожидает окончания выполнения команды. !ВНИМАНИЕ! Логирование в такой ситуации будет невозможно; False - Агент ожидает окончания выполнения операции.
- **inSendOutputToOrchestratorLogsBool** – True - отправлять весь вывод от команды в логи Оркестратора; Flase - Не отправлять; Default True
- **inCMDEncodingStr** – Кодировка DOS среды, в которой исполняется команда. Если некорректно установить кодировку - русские символы будут испорчены. По умолчанию установлена «cp1251»
- **inCaptureBool** – True - не запускать приложение как отдельное. Результат выполнения команды будет выводиться в окне Агента (если окно Агента присутствует на экране). False - команда будет запущена в отдельном DOS окне.

Результат:

ГУИД (GUID) строка Активности (ActivityItem). Далее можно ожидать результат этой функции по ГУИД с помощью функции AgentActivityItemReturnGet

---

pyOpenRPA.Orchestrator.__Orchestrator__.AgentOSFileBinaryDataBase64StrAppend(*inHostNameStr, inUserStr, inFilePathStr, inFileDataBase64Str, inGSettings=None*) [исходный код]

L+,W+: Добавить бинарную информацию в существующий бинарный файл, который будет расположен по адресу inFilePathStr на стороне Агента с содержимым, декодированным с формата base64: inFileDataBase64Str

Параметры:

- **inHostNameStr** – Наименование хоста, на котором запущен Агент. Наименования подключенных агентов доступно для просмотра в панели управления
- **inUserStr** – Наименование пользователя, на графической сессии которого запущен Агент. Наименования подключенных агентов доступно для просмотра в панели управления
- **inFilePathStr** – Полный путь к сохраняемому файлу на стороне Агента.
- **inFileDataBase64Str** – Строка в формате base64 для отправки в создаваемый файл на стороне Агента.
- **inGSettings** – Глобальный словарь настроек Оркестратора (синглтон)

Результат:

ГУИД (GUID) строка Активности (ActivityItem). Далее можно ожидать результат этой функции по ГУИД с помощью функции AgentActivityItemReturnGet

---

pyOpenRPA.Orchestrator.__Orchestrator__.AgentOSFileBinaryDataBase64StrCreate(*inHostNameStr, inUserStr, inFilePathStr, inFileDataBase64Str, inGSettings=None*) [исходный код]

L+,W+: Создать бинарный файл, который будет расположен по адресу inFilePathStr на стороне Агента с содержимым, декодированным с формата base64: inFileDataBase64Str

Параметры:

- **inHostNameStr** – Наименование хоста, на котором запущен Агент. Наименования подключенных агентов доступно для просмотра в панели управления
- **inUserStr** – Наименование пользователя, на графической сессии которого запущен Агент. Наименования подключенных агентов доступно для просмотра в панели управления
- **inFilePathStr** – Полный путь к сохраняемому файлу на стороне Агента.
- **inFileDataBase64Str** – Строка в формате base64 для отправки в создаваемый файл на стороне Агента.

- **inGSettings** – Глобальный словарь настроек Оркестратора (синглтон)

Результат:
ГУИД (GUID) строка Активности (ActivityItem). Далее можно ожидать результат этой функции по ГУИД с помощью функции AgentActivityItemReturnGet

---

pyOpenRPA.Orchestrator.__Orchestrator__.AgentOSFileBinaryDataBase64StrReceive(*inHostNameStr, inUserStr, inFilePathStr, inGSettings=None*) [исходный код]

L+,W+: Выполнить чтение бинарного файла и получить содержимое в формате base64 (строка)

Параметры:
- **inHostNameStr** – Наименование хоста, на котором запущен Агент. Наименования подключенных агентов доступно для просмотра в панели управления
- **inUserStr** – Наименование пользователя, на графической сессии которого запущен Агент. Наименования подключенных агентов доступно для просмотра в панели управления
- **inFilePathStr** – Путь к бинарному файлу на чтение на стороне Агента
- **inGSettings** – Глобальный словарь настроек Оркестратора (синглтон)

Результат:
ГУИД (GUID) строка Активности (ActivityItem). Далее можно ожидать результат этой функции по ГУИД с помощью функции AgentActivityItemReturnGet

---

pyOpenRPA.Orchestrator.__Orchestrator__.AgentOSFileBinaryDataBytesCreate(*inHostNameStr, inUserStr, inFilePathStr, inFileDataBytes, inGSettings=None*) [исходный код]

L+,W+: Создать бинарный файл, который будет расположен по адресу inFilePathStr на стороне Агента с содержимым inFileDataBytes

Параметры:
- **inHostNameStr** – Наименование хоста, на котором запущен Агент. Наименования подключенных агентов доступно для просмотра в панели управления
- **inUserStr** – Наименование пользователя, на графической сессии которого запущен Агент. Наименования подключенных агентов доступно для просмотра в панели управления
- **inFilePathStr** – Полный путь к сохраняемому файлу на стороне Агента.
- **inFileDataBytes** – Строка байт (b"") для отправки в создаваемый файл на стороне Агента.
- **inGSettings** – Глобальный словарь настроек Оркестратора (синглтон)

Результат:
ГУИД (GUID) строка Активности (ActivityItem). Далее можно ожидать результат этой функции по ГУИД с помощью функции AgentActivityItemReturnGet

---

pyOpenRPA.Orchestrator.__Orchestrator__.AgentOSFileBinaryDataReceive(*inHostNameStr, inUserStr, inFilePathStr*) [исходный код]

L+,W+: Чтение бинарного файла на стороне Агента по адресу inFilePathStr.

!ВНИМАНИЕ - СИНХРОННАЯ! Функция не завершится, пока не будет получен результат чтения на стороне Агента.

Параметры:
- **inHostNameStr** – Наименование хоста, на котором запущен Агент. Наименования подключенных агентов доступно для просмотра в панели управления
- **inUserStr** – Наименование пользователя, на графической сессии которого запущен Агент. Наименования подключенных агентов доступно для просмотра в панели управления
- **inFilePathStr** – Путь к бинарному файлу, который требуется прочитать на стороне Агента

Результат:

Строка байт (b“”) - содержимое бинарного файла

pyOpenRPA.Orchestrator.__Orchestrator__.AgentOSFileSend(*inHostNameStr, inUserStr, inOrchestratorFilePathStr, inAgentFilePathStr, inGSettings=None*) [исходный код]

L+,W+: Отправить файл по адресу inOrchestratorFilePathStr со стороны Оркестратора и сохранить по адресу inAgentFilePathStr на стороне Агента. Поддерживает передачу крупных файлов (более 2-х Гб.). Функция является синхронной - не закончит свое выполнение, пока файл не будет передан полностью.

!ВНИМАНИЕ - ПОТОКОБЕЗОПАСНАЯ! Вы можете вызвать эту функцию до инициализации ядра Оркестратора. Оркестратор добавит эту функцию в процессорную очередь на исполение. Если вам нужен результат функции, то необходимо сначала убедиться в том, что ядро Оркестратора было инициализированно (см. функцию OrchestratorInitWait).

Параметры:

- **inHostNameStr** – Наименование хоста, на котором запущен Агент. Наименования подключенных агентов доступно для просмотра в панели управления
- **inUserStr** – Наименование пользователя, на графической сессии которого запущен Агент. Наименования подключенных агентов доступно для просмотра в панели управления
- **inOrchestratorFilePathStr** – Полный путь к передаваемому файлу на стороне Оркестратора.
- **inAgentFilePathStr** – Полный путь к локации, в которую требуется сохранить передаваемый файл.
- **inGSettings** – Глобальный словарь настроек Оркестратора (синглтон)

Результат:

ГУИД (GUID) строка Активности (ActivityItem). Далее можно ожидать результат этой функции по ГУИД с помощью функции AgentActivityItemReturnGet

pyOpenRPA.Orchestrator.__Orchestrator__.AgentOSFileTextDataStrCreate(*inHostNameStr, inUserStr, inFilePathStr, inFileDataStr, inEncodingStr='utf-8', inGSettings=None*) [исходный код]

L+,W+: Создать текстовый файл, который будет расположен по адресу inFilePathStr на стороне Агента с содержимым inFileDataStr в кодировке inEncodingStr

Параметры:

- **inHostNameStr** – Наименование хоста, на котором запущен Агент. Наименования подключенных агентов доступно для просмотра в панели управления
- **inUserStr** – Наименование пользователя, на графической сессии которого запущен Агент. Наименования подключенных агентов доступно для просмотра в панели управления
- **inFilePathStr** – Полный путь к сохраняемому файлу на стороне Агента.
- **inFileDataStr** – Строка для отправки в создаваемый файл на стороне Агента.
- **inEncodingStr** – Кодировка текстового файла. По умолчанию utf-8
- **inGSettings** – Глобальный словарь настроек Оркестратора (синглтон)

Результат:

ГУИД (GUID) строка Активности (ActivityItem). Далее можно ожидать результат этой функции по ГУИД с помощью функции AgentActivityItemReturnGet

pyOpenRPA.Orchestrator.__Orchestrator__.AgentOSFileTextDataStrReceive(*inHostNameStr, inUserStr, inFilePathStr, inEncodingStr='utf-8', inGSettings=None*) [исходный код]

L+,W+: Чтение текстового файла на стороне Агента по адресу inFilePathStr. По ГИУД с помощью функции AgentActivityItemReturnGet можно будет получить текстовую строку данных, которые были расположены в файле.

!ВНИМАНИЕ - АСИНХРОННАЯ! Функция завершится сразу, не дожидаясь окончания выполнения операции на стороне Агента.

Параметры:

- **inHostNameStr** – Наименование хоста, на котором запущен Агент. Наименования подключенных агентов доступно для просмотра в панели управления
- **inUserStr** – Наименование пользователя, на графической сессии которого запущен Агент. Наименования подключенных агентов доступно для просмотра в панели управления
- **inFilePathStr** – Путь к бинарному файлу, который требуется прочитать на стороне Агента
- **inEncodingStr** – Кодировка текстового файла. По умолчанию utf-8
- **inGSettings** – Глобальный словарь настроек Оркестратора (синглтон)

Результат:

ГУИД (GUID) строка Активности (ActivityItem). Далее можно ожидать результат этой функции по ГУИД с помощью функции AgentActivityItemReturnGet

pyOpenRPA.Orchestrator.__Orchestrator__.AgentOSLogoff(*inHostNameStr, inUserStr*) [исходный код]

L+,W+: Выполнить операцию logoff на стороне пользователя.

Параметры:

- **inHostNameStr** – Наименование хоста, на котором запущен Агент. Наименования подключенных агентов доступно для просмотра в панели управления
- **inUserStr** – Наименование пользователя, на графической сессии которого запущен Агент. Наименования подключенных агентов доступно для просмотра в панели управления

Результат:

ГУИД (GUID) строка Активности (ActivityItem). Далее можно ожидать результат этой функции по ГУИД с помощью функции AgentActivityItemReturnGet

pyOpenRPA.Orchestrator.__Orchestrator__.AgentProcessWOExeUpperUserListGet(*inHostNameStr, inUserStr, inGSettings=None*) [исходный код]

L-,W+: Получить список процессов, которые выполняется на сессии Агента. Все процессы фиксируются без постфикса .exe, а также в верхнем регистре.

ПРИМЕР РЕЗУЛЬТАТА, КОТОРЫЙ МОЖНО ПОЛУЧИТЬ ПО ГУИД ЧЕРЕЗ ФУНКЦИЮ AgentActivityItemReturnGet: [«ORCHESTRATOR», «AGENT», «CHROME», «EXPLORER», ...]

Параметры:

- **inGSettings** – Глобальный словарь настроек Оркестратора (синглтон)
- **inHostNameStr** – Наименование хоста, на котором запущен Агент. Наименования подключенных агентов доступно для просмотра в панели управления
- **inUserStr** – Наименование пользователя, на графической сессии которого запущен Агент. Наименования подключенных агентов доступно для просмотра в панели управления

Результат:

ГУИД (GUID) строка Активности (ActivityItem). Далее можно ожидать результат этой функции по ГУИД с помощью функции AgentActivityItemReturnGet

pyOpenRPA.Orchestrator.__Orchestrator__.GSettingsGet(*inGSettings=None*) [исходный код]

L+,W+: Вернуть глобальный словарь настроек Оркестратора. Во время выполнения программы глобальный словарь настроек существует в единственном экземпляре (синглтон)

Параметры:

inGSettings – Дополнительный словарь настроек, который необходимо добавить в основной глобальный словарь настроек Оркестратора (синглтон)

Результат:

Глобальный словарь настроек GSettings

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

L+,W+: Применить операцию .append к обьекту, расположенному по адресу списка ключей inKeyList в глобальном словаре настроек Оркестратора GSettings. Пример: Добавить значение в конец списка (list).

```
# ПРИМЕР
from pyOpenRPA import Orchestrator

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

Параметры:

- **inGSettings** – Глобальный словарь настроек Оркестратора (синглтон)
- **inValue** – Значение для установки в глобальный словарь настроек Оркестратора GSettings
- **inKeyList** – Список ключей, по адресу которого выполнить добавление в конец списка (list)

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

L+,W+: Получить значение из глобального словаря настроек Оркестратора GSettings по списку ключей.

Параметры:

- **inGSettings** – Глобальный словарь настроек Оркестратора (синглтон)
- **inKeyList** – Список ключей, по адресу которого получить значение из GSettings

Результат:

Значение (тип данных определяется форматом хранения в GSettings)

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

L+,W+: Применить оператор сложения (+) к обьекту, расположенному по адресу списка ключей inKeyList в глобальном словаре настроек Оркестратора GSettings. Пример: соединить 2 списка (list).

```
# ПРИМЕР
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
#        ]
#    }
#}
```

Параметры:

- **inGSettings** – Глобальный словарь настроек Оркестратора (синглтон)
- **inValue** – Значение для установки в глобальный словарь настроек Оркестратора GSettings
- **inKeyList** – Список ключей, по адресу которого выполнить добавление в конец списка (list)

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

L+,W+: Установить значение из глобального словаря настроек Оркестратора GSettings по списку ключей.

Пример: Для того, чтобы установить значение для ключа car в словаре {«complex»:{«car»:»green»}, «simple»:»HELLO»}, необходимо передать список ключей: [«complex», «car»]

Параметры:

- **inGSettings** – Глобальный словарь настроек Оркестратора (синглтон)
- **inValue** – Значение для установки в глобальный словарь настроек Оркестратора GSettings
- **inKeyList** – Список ключей, по адресу которого установить значение в GSettings

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

L-,W+: Отправить команду на выполнение на сессию, где выполняется Оркестратор.

Параметры:

- **inCMDStr** – Команда на отправку
- **inRunAsyncBool** – True - выполнить команду в асинхронном режиме (не дожидаться окончания выполнения программы и не захватывать результат выполнения); False - Ждать окончания выполнения и захватывать результат
- **inLogger** – Логгер, в который отправлять информацию о результате выполнения команды

Результат:

Строка результата выполнения команды. Если inRunAsyncBool = False

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

L+,W+: Выполнить верификацию доменного (локального) пользователя по паре логин/пароль

Параметры:

- **inUserStr** – Наименование пользователя
- **inPasswordStr** – Пароль
- **inDomainStr** – Домен. Если домена нет - не указывать или «»

Результат:

True - Учетные данные верны; False - Учетные данные представлены некорректно

pyOpenRPA.Orchestrator.__Orchestrator__.OSLogoff() [исходный код]

L+,W+: Выполнить отключение сессии, на которой выполняется Оркестратор.

Результат:

pyOpenRPA.Orchestrator.__Orchestrator__.OSRemotePCRestart(*inHostStr*, *inForceBool=True*, *inLogger=None*) [исходный код]

L-,W+: Отправить сигнал на удаленную перезагрузку операционной системы.

!ВНИМАНИЕ! Перезапуск будет принят, если учетная запись имеет полномочия на перезапуск на соответсвующей машине.

Параметры:

- **inHostStr** – Имя хоста, который требуется перезагрузить
- **inForceBool** – True - принудительная перезагрузка; False - мягкая перезагрузка (дождаться окончания выполнения всех операций). По умолчанию True
- **inLogger** – Логгер, в который отправлять информацию о результате выполнения команды

Результат:

pyOpenRPA.Orchestrator.__Orchestrator__.OSRestart(*inForceBool=True, inLogger=None*) [исходный код]

L+,W+: Отправить сигнал на перезагрузку операционной системы.

!ВНИМАНИЕ! Перезапуск будет принят, если учетная запись имеет полномочия на перезапуск на соответсвующей машине.

Параметры:

- **inForceBool** – True - принудительная перезагрузка; False - мягкая перезагрузка (дождаться окончания выполнения всех операций). По умолчанию True
- **inLogger** – Логгер, в который отправлять информацию о результате выполнения команды

Результат:

pyOpenRPA.Orchestrator.__Orchestrator__.Orchestrator(*inGSettings=None, inDumpRestoreBool=True, inRunAsAdministratorBool=True*) [исходный код]

L+,W+: Инициализация ядра Оркестратора (всех потоков)

Параметры:

- **inGSettings** – Глобальный словарь настроек Оркестратора (синглтон)
- **inDumpRestoreBool** – True - Восстановить информацию о RDP сессиях и StorageDict; False - не восстанавливать
- **inRunAsAdministratorBool** – True - Проверить права администратратора и обеспечить их; False - Не обеспечивать права администратора

pyOpenRPA.Orchestrator.__Orchestrator__.OrchestratorInitWait()→ None [исходный код]

L+,W+: Ожидать инициализацию ядра Оркестратора

!ВНИМАНИЕ!: НИ В КОЕМ СЛУЧАЕ НЕ ЗАПУСКАТЬ ЭТУ ФУНКЦИЮ В ОСНОВНОМ ПОТОКЕ, ГДЕ ПРОИСХОДИТ ИНИЦИАЛИЗАЦИЯ ЯДРА ОРКЕСТРАТОРА - ВОЗНИКНЕТ ВЕЧНЫЙ ЦИКЛ

pyOpenRPA.Orchestrator.__Orchestrator__.OrchestratorIsAdmin() [исходный код]

L+,W+: Проверить, запущен ли Оркестратор с правами администратора. Права администратора нужны Оркестратору только для контроля графической сессии, на которой он запущен. Если эти права выделить индивидуально, то права администратора будут необязательны.

Результат:

True - Запущен с правами администратора; False - Не запущен с правами администратора

pyOpenRPA.Orchestrator.__Orchestrator__.OrchestratorIsCredentialsAsk() [исходный код]

L+,W+: Проверить, активирована ли авторизация при переходе к Оркестратору.

Результат:

True - Активирована; False - Деактивирована

pyOpenRPA.Orchestrator.__Orchestrator__.OrchestratorIsInited()→ bool [исходный код]

L+,W+: Проверить, было ли проинициализировано ядро Оркестратора

Результат:

True - Ядро Оркестратора было проинициализировано; False - Требуется время на инициализацию

Тип результата:

bool

pyOpenRPA.Orchestrator.__Orchestrator__.OrchestratorLoggerGet()→ Logger [исходный код]

L+,W+: Получить логгер Оркестратора

Результат:

Логгер

pyOpenRPA.Orchestrator.__Orchestrator__.OrchestratorPySearchInit(*inGlobPatternStr*, *inDefStr=None*, *inDefArgNameGSettingsStr=None*, *inAsyncInitBool=False*, *inPackageLevelInt=0*) [исходный код]

L+,W+: Выполнить поиск и инициализацию пользовательских .py файлов в Оркестраторе (например панелей управления роботов)

Добавляет инициализированный модуль в пространство sys.modules как imported (имя модуля = имя файла без расширения).

ВНИМАНИЕ! ПРЕОБРАЗУЕТ ПУТИ МЕЖДУ WINDOWS И LINUX НОТАЦИЯМИ

```
# ВАРИАНТ ИСПОЛЬЗОВАНИЯ 1 (инициализация модуля py без вызова каких-либо функций внутри)
# автоинициализация всех .py файлов, с префиксом CP_, которые расположены в папке ControlPanel
Orchestrator.OrchestratorPySearchInit(inGlobPatternStr="ControlPanel\CP_*.py")

# ВАРИАНТ ИСПОЛЬЗОВАНИЯ 2 (инициализация модуля py с вызовом функции внутри) - преимущественно для обр
# автоинициализация всех .py файлов, с префиксом CP_, которые расположены в папке ControlPanel
Orchestrator.OrchestratorPySearchInit(inGlobPatternStr="ControlPanel\CP_*.py", inDefStr="SettingsUpdat

# ДЛЯ СПРАВКИ & ИСТОРИИ: Код выше позволил отказаться от блока кода ниже для каждого .py файла
## !!! 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.")
```

Параметры:

- **inGlobPatternStr** – Пример «..***_CP*.py»
- **inDefStr** – ОПЦИОНАЛЬНО Строковое наименование функции. Преимущественно для обратной совместимости
- **inDefArgNameGSettingsStr** – ОПЦИОНАЛЬНО Наименование аргумента, в который требуется передать GSettings (если необходимо)
- **inAsyncInitBool** – ОПЦИОНАЛЬНО True - Инициализация py модулей в отдельных параллельных потоках - псевдопараллельное выполнение. False - последовательная инициализация
- **inPackageLevelInt** – ОПЦИОНАЛЬНО Уровень вложенности модуля в пакет. По умолчанию 0 (не является модулем пакета)

Результат:

Сведения об инициализированных модулях в структуре: { «ModuleNameStr»:{«PyPathStr»: «», «Module»: ...}, ...}

pyOpenRPA.Orchestrator.__Orchestrator__.OrchestratorRerunAsAdmin() [исходный код]

L-,W+: Перезапустить Оркестратор с правами локального администратора. Права администратора нужны Оркестратору только для контроля графической сессии, на которой он

запущен. Если эти права выделить индивидуально, то права администратора будут необязательны.

Результат:
True - Запущен с правами администратора; False - Не запущен с правами администратора

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

L+,W+: Перезапуск Оркестратора с сохранением информации о запущенных RDP сессиях.

Параметры:
inGSettings – Глобальный словарь настроек Оркестратора (синглтон)

pyOpenRPA.Orchestrator.__Orchestrator__.OrchestratorScheduleGet()→ <module 'schedule' from 'D:\\RPA\\ORPA\\Resources\\WPy64-3720\\python-3.7.2.amd64\\lib\\site-packages\\schedule__init__.py'> [исходный код]

L+,W+: Базовый объект расписания, который можно использовать для запуска / остановки роботов. Подробнее про объект schedule и его примеры использования см. по адресу: schedule.readthedocs.io

```
# Однопоточный schedule
Orchestrator.OrchestratorScheduleGet().every(5).seconds.do(lProcess.StatusCheckStart)

#Многопоточный schedule. см. описание Orchestrator.OrchestratorThreadStart
Orchestrator.OrchestratorScheduleGet().every(5).seconds.do(Orchestrator.OrchestratorThreadStart,lProce
```

Результат:
schedule объект

pyOpenRPA.Orchestrator.__Orchestrator__.OrchestratorSessionRestore(*inGSettings=None*) [исходный код]

L+,W+: Восстановить состояние Оркестратора, если ранее состояние Оркестратора было сохранено с помощью функции OrchestratorSessionSave:

- RDP сессий, которые контролирует Оркестратор
- Хранилища DataStorage в глобальном словаре настроек GSettings. DataStorage поддерживает хранение объектов Python

(до версии 1.2.7)

_SessionLast_GSettings.pickle (binary)

(начиная с версии 1.2.7)

_SessionLast_RDPList.json (encoding = «utf-8») _SessionLast_StorageDict.pickle (binary)

Параметры:
inGSettings – Глобальный словарь настроек Оркестратора (синглтон)

pyOpenRPA.Orchestrator.__Orchestrator__.OrchestratorSessionSave(*inGSettings=None*) [исходный код]

L+,W+: Сохранить состояние Оркестратора (для дальнейшего восстановления в случае перезапуска):

- RDP сессий, которые контролирует Оркестратор
- Хранилища DataStorage в глобальном словаре настроек GSettings. DataStorage поддерживает хранение объектов Python

(до версии 1.2.7)

_SessionLast_GSettings.pickle (binary)

(начиная с версии 1.2.7)

_SessionLast_RDPList.json (encoding = «utf-8») _SessionLast_StorageDict.pickle (binary)

Параметры:

inGSettings – Глобальный словарь настроек Оркестратора (синглтон)

---

pyOpenRPA.Orchestrator.__Orchestrator__.OrchestratorThreadStart(*inDef*, **inArgList*, ***inArgDict*) [исходный код]

L+,W+: Запустить функцию в отдельном потоке. В таком случае получить результат выполнения функции можно только через общие переменные. (Например через GSettings)

Параметры:

- **inDef** – Python функция
- **inArgList** – Список неименованных аргументов функции inDef
- **inArgDict** – Словарь именованных аргументов функции inDef

Результат:

threading.Thread экземпляр

---

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

L+,W+: Периодический вызов функции Python.

Параметры:

- **inDef** – Функция Python, которую потребуется периодически вызывать
- **inIntervalSecFloat** – Интервал между вызовами функции в сек.
- **inIntervalAsyncBool** – False - ожидать интервал inIntervalSecFloat только после окончания выполнения предыдущей итерации; True - Ожидать интервал сразу после запуска итерации
- **inDefArgList** – Список (list) неименованных аргументов для передачи в функцию. По умолчанию None
- **inDefArgDict** – Словарь (dict) именованных аргументов для передачи в функцию. По умолчанию None
- **inDefArgGSettingsNameStr** – Наименование аргумента глобального словаря настроек Оркестратора GSettings (опционально)
- **inDefArgLoggerNameStr** – Наименование аргумента логгера Оркестратора (опционально)
- **inExecuteInNewThreadBool** – True - периодический вызов выполнять в отдельном потоке (не останавливать выполнение текущего потока); False - Выполнение периодического вызова в текущем потоке, в котором была вызвана функция ProcessDefIntervalCall. По умолчанию: True
- **inLogger** – Логгер для фиксации сообщений выполнения функции (опционально)
- **inGSettings** – Глобальный словарь настроек Оркестратора (синглтон)

---

pyOpenRPA.Orchestrator.__Orchestrator__.ProcessIsStarted(*inProcessNameWOExeStr*) [исходный код]

L-,W+: Проверить, запущен ли процесс, который в наименовании содержит inProcessNameWOExeStr.

!ВНИМАНИЕ! ПРОВЕРЯЕТ ВСЕ ПРОЦЕССЫ ОПЕРАЦИОННОЙ СИСТЕМЫ. И ДРУГИХ ПОЛЬЗОВАТЕЛЬСКИХ СЕССИЙ, ЕСЛИ ОНИ ЗАПУЩЕНЫ НА ЭТОЙ МАШИНЕ.

```
# ПРИМЕР
from pyOpenRPA import Orchestrator

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

Параметры:

inProcessNameWOExeStr – Наименование процесса без расширения .exe (WO = WithOut = Без) Пример: Для проверки процесса блокнота нужно указывать «notepad», не «notepad.exe»

Результат:

True - Процесс запущен на ОС ; False - Процесс не запущен на ОС, где работает Оркестратор

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

L-,W+: Вернуть список процессов, запущенных на ОС, где работает Оркестратор. В списке отсортировать процессы по количеству используемой оперативной памяти. Также можно указать перечень процессов, которые интересуют - функция будет показывать активные из них.

!ВНИМАНИЕ! ДЛЯ ПОЛУЧЕНИЯ СПИСКА ВСЕХ ПРОЦЕССОВ ОС НЕОБХОДИМО ЗАПУСКАТЬ ОРКЕСТРАТОР С ПРАВАМИ АДМИНИСТРАТОРА.

```
# ПРИМЕР
from pyOpenRPA import Orchestrator

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

Параметры:

inProcessNameWOExeList – Список процессов, среди которых искать активные. Если параметр не указывать - вернет перечень всех доступных процессов

Результат:

Сведения о запущенных процессах в следующем формате { «ProcessWOExeList»: [«notepad»,»...»], «ProcessWOExeUpperList»: [«NOTEPAD»,»...»], «ProcessDetailList»: [

```
{
```

„pid“: 412, # Идентификатор процесса „username“: «DESKTOPUSER», „name“: „notepad.exe“, „vms“: 13.77767775, # В Мб „NameWOExeUpperStr“: „NOTEPAD“, „NameWOExeStr“: «„notepad“»},

{...}]

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

L-,W+: Запуск процесса на сессии Оркестратора, если на ОС не запущен процесс inStopProcessNameWOExeStr.

!ВНИМАНИЕ! ПРИ ПРОВЕРКЕ РАНЕЕ ЗАПУЩЕННЫХ ПРОЦЕССОВ ПРОВЕРЯЕТ ВСЕ ПРОЦЕССЫ ОПЕРАЦИОННОЙ СИСТЕМЫ. И ДРУГИХ ПОЛЬЗОВАТЕЛЬСКИХ СЕССИЙ, ЕСЛИ ОНИ ЗАПУЩЕНЫ НА ЭТОЙ МАШИНЕ.

```
# ПРИМЕР
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
```

Параметры:

- **inPathStr** – Путь к файлу запускаемого процесса
- **inArgList** – Список аргументов, передаваемых в запускаемый процесс. Пример: [«test.txt»]
- **inStopProcessNameWOExeStr** – Доп. контроль: Не запускать процесс, если обнаружен запущенный процесс под наименованием inStopProcessNameWOExeStr без расширения .exe (WO = WithOut = Без)

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

L-,W+: Остановить процесс на ОС, где работает Оркестратор, под учетной записью inUserNameStr.

```
# ПРИМЕР
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
```

Параметры:

- **inProcessNameWOExeStr** – Наименование процесса без расширения .exe (WO = WithOut = Без). Пример: Для проверки процесса блокнота нужно указывать «notepad», не «notepad.exe»
- **inCloseForceBool** – True - принудительно завершить процесс. False - отправить сигнал на безопасное отключение (!ВНИМАНИЕ! - ОС не позволяет отправлять сигнал на безопасное отключение на другую сесиию - только на той, где работает Оркестратор)
- **inUserNameStr** – Наименование пользователя, под именем которого искать процесс для остановки. По умолчанию «%username%» - искать процесс на текущей сессии

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

L+,W+: Добавить активность (ActivityItem) в процессорную очередь.

```
# USAGE
from pyOpenRPA import Orchestrator

# EXAMPLE 1
def TestDef(inArg1Str, inGSettings, inLogger):
    pass
lActivityItem = 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")
lActivityItem = Orchestrator.ProcessorActivityItemCreate(
    inDef = "TestDefAlias",
    inArgList=[],
    inArgDict={"inArg1Str": "ArgValueStr"},
    inArgGSettingsStr = None,
    inArgLoggerStr = None)
Orchestrator.ProcessorActivityItemAppend(
    inGSettings = gSettingsDict,
    inActivityItemDict = lActivityItem)
# Activity have been already append in the processor queue
```

Параметры:

- **inGSettings** – Глобальный словарь настроек Оркестратора (синглтон)
- **inDef** – Функция Python или синоним (текстовый ключ)
- **inArgList** – Список (list) неименованных аргументов к функции
- **inArgDict** – Словарь (dict) именованных аргументов к функции
- **inArgGSettingsStr** – Текстовое наименование аргумента GSettings (если требуется передавать)
- **inArgLoggerStr** – Текстовое наименование аргумента logger (если требуется передавать)
- **inGUIDStr** – ГУИД идентификатор активности (ActivityItem). Если ГУИД не указан, то он будет сгенерирован автоматически
- **inThreadBool** – True - выполнить ActivityItem в новом потоке; False - выполнить последовательно в общем потоке процессорной очереди
- **inActivityItemDict** – Альтернативный вариант заполнения, если уже имеется Активность (ActivityItem). В таком случае не требуется заполнять аргументы inDef, inArgList, inArgDict, inArgGSettingsStr, inArgLoggerStr

:return ГУИД активности (ActivityItem)

pyOpenRPA.Orchestrator.__Orchestrator__.ProcessorActivityItemCreate(*inDef*, *inArgList=None*, *inArgDict=None*, *inArgGSettingsStr=None*, *inArgLoggerStr=None*, *inGUIDStr=None*, *inThreadBool=False*) [исходный код]

L+,W+: Создать Активность (ActivityItem). Активность можно использовать в ProcessorActivityItemAppend или в Processor.ActivityListExecute или в функциях работы с Агентами.

Старая версия. Новую версию см. в ActivityItemCreate

```
# ПРИМЕР
from pyOpenRPA import Orchestrator

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

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

Параметры:

- **inDef** – Функция Python или синоним (текстовый ключ)
- **inArgList** – Список (list) неименованных аргументов к функции
- **inArgDict** – Словарь (dict) именованных аргументов к функции
- **inArgGSettingsStr** – Текстовое наименование аргумента GSettings (если требуется передавать)
- **inArgLoggerStr** – Текстовое наименование аргумента logger (если требуется передавать)
- **inGUIDStr** – ГУИД идентификатор активности (ActivityItem). Если ГУИД не указан, то он будет сгенерирован автоматически
- **inThreadBool** – True - выполнить ActivityItem в новом потоке; False - выполнить последовательно в общем потоке процессорной очереди

Результат:

lActivityItemDict= {

«Def»:inDef, # def link or def alias (look gSettings[«Processor»][«AliasDefDict»]) «ArgList»:inArgList, # Args list «ArgDict»:inArgDict, # Args dictionary «ArgGSettings»: inArgGSettingsStr, # Name of GSettings attribute: str (ArgDict) or index (for ArgList) «ArgLogger»: inArgLoggerStr, # Name of GSettings attribute: str (ArgDict) or index (for ArgList) «GUIDStr»: inGUIDStr, «ThreadBool»: inThreadBool

}

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

L+,W+: Создать синоним (текстовый ключ) для инициации выполнения функции в том случае, если запрос на выполнения пришел из вне (передача функций невозможна).

Старая версия. Новую версию см. ActivityItemDefAliasCreate

```
# 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 = lAlias
```

Параметры:

- **inDef** – функция Python
- **inAliasStr** – Строковый ключ (синоним), который можно будет использовать в Активности (ActivityItem)
- **inGSettings** – Глобальный словарь настроек Оркестратора (синглтон)

Результат:

Строковый ключ, который был назначен. Ключ может быть изменен, если входящий текстовый ключ был уже занят.

pyOpenRPA.Orchestrator.__Orchestrator__.ProcessorAliasDefUpdate(*inDef, inAliasStr, inGSettings=None*) [исходный код]

L+,W+: Обновить синоним (текстовый ключ) для инициации выполнения функции в том случае, если запрос на выполнения пришел из вне (передача функций невозможна).

Старая версия. Новую версию см. ActivityItemDefAliasUpdate

```
# 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 = "TestDefAl
```

Параметры:

- **inGSettings** – Глобальный словарь настроек Оркестратора (синглтон)
- **inDef** – функция Python
- **inAliasStr** – Строковый ключ (синоним), который можно будет использовать в Активности (ActivityItem)

Результат:

Строковый ключ, который был назначен. Ключ будет тем же, если входящий текстовый ключ был уже занят.

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

L+,W+: Импорт модуля и выполнение функции в процессе Оркестратора.

Примечание

Импорт модуля inModulePathStr будет происходить каждый раз в вызовом функции PythonStart.

```
# ПРИМЕР
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"
```

Параметры:

- **inModulePathStr** – Абсолютный или относительный путь (относительно рабочей директории процесса Оркестратора), по которому расположен импортируемый .py файл
- **inDefNameStr** – Наименование функции в модуле .py
- **inArgList** – Список (list) неименованных аргументов для передачи в функцию. По умолчанию None
- **inArgDict** – Словарь (dict) именованных аргументов для передачи в функцию. По умолчанию None
- **inLogger** – Логгер для фиксации сообщений выполнения функции (опционально)

pyOpenRPA.Orchestrator.__Orchestrator__.RDPSessionCMDRun(*inRDPSessionKeyStr*, *inCMDStr*, *inModeStr='CROSSCHECK'*, *inGSettings=None*) [исходный код]

L-,W+: Отправить CMD команду на удаленную сесиию через RDP окно (без Агента).

!ВНИМАНИЕ! Данная функция может работать нестабильно из-за использования графических элементов UI при работе с RDP. Мы рекомендуем использовать конструкцию взаимодействия Оркестратора с Агентом.

```
# ПРИМЕР
from pyOpenRPA import Orchestrator

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

Параметры:

- **inRDPSessionKeyStr** – Ключ RDP сессии - необходим для дальнейшей идентификации
- **inCMDStr** – Команда CMD, которую отправить на удаленную сессию
- **inModeStr** – По умолчанию «CROSSCHECK». Варианты: «LISTEN» - Получить результат выполнения операции. Внимание! необходим общий буфер обмена с сессией Оркестратора! «CROSSCHECK» - Выполнить проверку, что операция была выполнена (без получения результата отработки CMD команды). Внимание! необходим общий буфер обмена с сессией Оркестратора! «RUN» - Не получать результат и не выполнять проверку
- **inGSettings** – Глобальный словарь настроек Оркестратора (синглтон)

Результат:

Результат выполнения операции в соответсвии с параметрами инициализации функции.
Выходная структура:

{

«OutStr»: <> # Результат выполнения CMD (если inModeStr = «LISTEN»)

«IsResponsibleBool»: True|False # True - RDP приняло команду; False - обратная связь не была получена (если inModeStr = «CROSSCHECK»)

}

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

L-,W+: Выполнить подключение к RDP и следить за стабильностью соединения со стороны Оркестратора. !ВНИМАНИЕ! - Подключение будет проигнорировано, если соединение по таком RDP ключу уже было инициализировано ранее.

2 способа использования функции: ВАРИАНТ 1 (ОСНОВНОЙ): inGSettings, inRDPSessionKeyStr, inRDPTemplateDict. Для получения inRDPTemplateDict см. функцию Orchestrator.RDPTemplateCreate ВАРИАНТ 2 (ОБРАТНАЯ СОВМЕСТИМОСТЬ ДО ВЕРСИИ 1.1.20): inGSettings, inRDPSessionKeyStr, inHostStr, inPortStr, inLoginStr, inPasswordStr

```
# ПРИМЕР (ВАРИАНТ 1)
from pyOpenRPA import Orchestrator

lRDPItemDict = 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 = lRDPItemDict)
# Orchestrator will create RDP session by the lRDPItemDict configuration
```

Параметры:

- **inRDPSessionKeyStr** – Ключ RDP сессии - необходим для дальнейшей идентификации
- **inRDPTemplateDict** – Конфигурационный словарь (dict) RDP подключения (см. функцию Orchestrator.RDPTemplateCreate)
- **inLoginStr** – Логин учетной записи, на которую будет происходить вход по RDP. Обратная совместимость до версии v 1.1.20. Мы рекомендуем использовать inRDPTemplateDict (см. функцию Orchestrator.RDPTemplateCreate)
- **inPasswordStr** – Пароль учетной записи, на которую будет происходить вход по RDP. !ВНИМАНИЕ! Пароль нигде не сохраняется - конфиденциальность обеспечена. Обратная совместимость до версии v 1.1.20. Мы рекомендуем использовать inRDPTemplateDict (см. функцию Orchestrator.RDPTemplateCreate)
- **inHostStr** – Имя хоста, к которому планируется подключение по RDP. Пример «77.77.22.22». Обратная совместимость до версии v 1.1.20. Мы рекомендуем использовать inRDPTemplateDict (см. функцию Orchestrator.RDPTemplateCreate)
- **inPortInt** – RDP порт, по которому будет происходить подключение. По умолчанию 3389 (стандартный порт RDP). Обратная совместимость до версии v 1.1.20. Мы рекомендуем использовать inRDPTemplateDict (см. функцию Orchestrator.RDPTemplateCreate)
- **inGSettings** – Глобальный словарь настроек Оркестратора (синглтон)
- **inRedirectClipboardBool** – True - Синхронизировать буфер обмена между сессией Оркестратора и удаленной RDP сессией; False - Не синхронизировать буфер обмена. По умолчанию True (в целях обратной совместимости). !ВНИМАНИЕ! Для учетных записей роботов мы рекомендуем не синхронизировать буфер обмена, так как это может привести к ошибкам роботов, которые работают с клавиатурой и буфером обмена

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

L-,W+: Выполнить отключение RDP сессии и прекратить мониторить его активность.

```
# ПРИМЕР
from pyOpenRPA import Orchestrator

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

Параметры:

- **inRDPSessionKeyStr** – Ключ RDP сессии - необходим для дальнейшей идентификации
- **inBreakTriggerProcessWOExeList** – Список процессов, наличие которых в диспетчере задач приведет к прерыванию задачи по остановке RDP соединения. Пример [«notepad»]
- **inGSettings** – Глобальный словарь настроек Оркестратора (синглтон)

pyOpenRPA.Orchestrator.__Orchestrator__.RDPSessionFileStoredRecieve(*inRDPSessionKeyStr, inRDPFilePathStr, inHostFilePathStr, inGSettings=None*) [исходный код]

L-,W+: Получение файла со стороны RDP сессии на сторону Оркестратора через UI инструменты RDP окна (без Агента).

!ВНИМАНИЕ! Данная функция может работать нестабильно из-за использования графических элементов UI при работе с RDP. Мы рекомендуем использовать конструкцию взаимодействия Оркестратора с Агентом.

!ВНИМАНИЕ! Для работы функции требуется открыть доступ по RDP к тем дискам, с которых будет производится копирование файла.

```
# ПРИМЕР
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)
```

Параметры:

- **inRDPSessionKeyStr** – Ключ RDP сессии - необходим для дальнейшей идентификации
- **inRDPFilePathStr** – Откуда скопировать файл. !Абсолютный! путь на стороне RDP сессии. Пример: «C:RPATESTDIRTest.py»
- **inHostFilePathStr** – Куда скопировать файл. Относительный или абсолютный путь к файлу на стороне Оркестратора. Пример: «TESTDIRTest.py»
- **inGSettings** – Глобальный словарь настроек Оркестратора (синглтон)

pyOpenRPA.Orchestrator.__Orchestrator__.RDPSessionFileStoredSend(*inRDPSessionKeyStr, inHostFilePathStr, inRDPFilePathStr, inGSettings=None*) [исходный код]

L-,W+: Отправка файла со стороны Оркестратора на сторону RDP сессии через UI инструменты RDP окна (без Агента).

!ВНИМАНИЕ! Данная функция может работать нестабильно из-за использования графических элементов UI при работе с RDP. Мы рекомендуем использовать конструкцию взаимодействия Оркестратора с Агентом.

!ВНИМАНИЕ! Для работы функции требуется открыть доступ по RDP к тем дискам, с которых будет производится копирование файла.

```
# ПРИМЕР
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)
```

Параметры:

- **inRDPSessionKeyStr** – Ключ RDP сессии - необходим для дальнейшей идентификации
- **inHostFilePathStr** – Откуда взять файл. Относительный или абсолютный путь к файлу на стороне Оркестратора. Пример: «TESTDIRTest.py»
- **inRDPFilePathStr** – Куда скопировать файл. !Абсолютный! путь на стороне RDP сессии. Пример: «C:RPATESTDIRTest.py»
- **inGSettings** – Глобальный словарь настроек Оркестратора (синглтон)

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

L-,W+: Выполнить отключение (logoff) на RDP сессии и прекратить мониторить активность со стороны Оркестратора.

```
# ПРИМЕР
from pyOpenRPA import Orchestrator

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

Параметры:

- **inRDPSessionKeyStr** – Ключ RDP сессии - необходим для дальнейшей идентификации
- **inBreakTriggerProcessWOExeList** – Список процессов, наличие которых в диспетчере задач приведет к прерыванию задачи по остановке RDP соединения. Пример [«notepad»]
- **inGSettings** – Глобальный словарь настроек Оркестратора (синглтон)

Результат:

True - Отключение прошло успешно; False - были зафиксированы ошибки при отключении.

pyOpenRPA.Orchestrator.__Orchestrator__.RDPSessionMonitorStop(*inRDPSessionKeyStr, inGSettings=None*) [исходный код]

L-,W+: Прекратить мониторить активность RDP соединения со стороны Оркестратора. Данная функция не уничтожает активное RDP соединение.

```
# ПРИМЕР
from pyOpenRPA import Orchestrator

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

Параметры:

- **inRDPSessionKeyStr** – Ключ RDP сессии - необходим для дальнейшей идентификации
- **inGSettings** – Глобальный словарь настроек Оркестратора (синглтон)

pyOpenRPA.Orchestrator.__Orchestrator__.RDPSessionProcessStartIfNotRunning(*inRDPSessionKeyStr, inProcessNameWEXEStr, inFilePathStr, inFlagGetAbsPathBool=True, inGSettings=None*) [исходный код]

L-,W+: Выполнить запуск процесса на RDP сессии через графические UI инструменты (без Агента).

!ВНИМАНИЕ! Данная функция может работать нестабильно из-за использования графических элементов UI при работе с RDP. Мы рекомендуем использовать конструкцию взаимодействия Оркестратора с Агентом.

```
# ПРИМЕР
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
```

Параметры:

- **inRDPSessionKeyStr** – Ключ RDP сессии - необходим для дальнейшей идентификации
- **inProcessNameWEXEStr** – Наименование процесса с расширением .exe (WEXE - WITH EXE - С EXE). Параметр позволяет не допустить повторного запуска процесса, если он уже был запущен. Example: «Notepad.exe»
- **inFilePathStr** – Путь к файлу запуска процесса на стороне RDP сессии
- **inFlagGetAbsPathBool** – True - Преобразовать относительный путь inFilePathStr в абсолютный с учетом рабочей директории Оркестратора; False - Не выполнять преобразований
- **inGSettings** – Глобальный словарь настроек Оркестратора (синглтон)

pyOpenRPA.Orchestrator.__Orchestrator__.RDPSessionProcessStop(*inRDPSessionKeyStr, inProcessNameWEXEStr, inFlagForceCloseBool, inGSettings=None*) [исходный код]

L-,W+: Отправка CMD команды в RDP окне на остановку процесса (без Агента).

!ВНИМАНИЕ! Данная функция может работать нестабильно из-за использования графических элементов UI при работе с RDP. Мы рекомендуем использовать конструкцию взаимодействия Оркестратора с Агентом.

```
# ПРИМЕР
from pyOpenRPA import Orchestrator

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

Параметры:

- **inRDPSessionKeyStr** – Ключ RDP сессии - необходим для дальнейшей идентификации
- **inProcessNameWEXEStr** – Наименование процесса, который требуется выключить с расширением .exe (WEXE - WITH EXE - С EXE). Пример: «Notepad.exe»
- **inFlagForceCloseBool** – True - Принудительное отключение. False - Отправка сигнала на безопасное отключение (если процесс поддерживает)
- **inGSettings** – Глобальный словарь настроек Оркестратора (синглтон)

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

L-,W+: Выполнить переподключение RDP сессии и продолжить мониторить его активность.

```
# ПРИМЕР
from pyOpenRPA import Orchestrator

lRDPItemDict = 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
```

Параметры:

- **inRDPSessionKeyStr** – Ключ RDP сессии - необходим для дальнейшей идентификации
- **inRDPTemplateDict** – Конфигурационный словарь (dict) RDP подключения (см. функцию Orchestrator.RDPTemplateCreate)
- **inGSettings** – Глобальный словарь настроек Оркестратора (синглтон)

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

L-,W+: Создать шаблон подключения RDP (dict). Данный шаблон далее можно использовать в Orchestrator.RDPSessionConnect

```
# ПРИМЕР
from pyOpenRPA import Orchestrator

lRDPItemDict = 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,
    inRedirectClipboardBool=False)
```

Параметры:

- **inLoginStr** – Логин учетной записи, на которую будет происходить вход по RDP
- **inPasswordStr** – Пароль учетной записи, на которую будет происходить вход по RDP. !ВНИМАНИЕ! Пароль нигде не сохраняется - конфиденциальность обеспечена
- **inHostStr** – Имя хоста, к которому планируется подключение по RDP. Пример «77.77.22.22»
- **inPortInt** – RDP порт, по которому будет происходить подключение. По умолчанию 3389 (стандартный порт RDP)
- **inWidthPXInt** – Разрешение ширины RDP графической сессии в пикселях. По умолчанию 1680
- **inHeightPXInt** – Разрешение высоты RDP графической сессии в пикселях. По умолчанию 1050
- **inUseBothMonitorBool** – True - Использовать все подключенные мониторы на RDP сессии; False - Использовать только один монитор на подключенной RDP сессии
- **inDepthBitInt** – Глубина цвета на удаленной RDP графической сессии. Допустимые варианты: 32 || 24 || 16 || 15. По умолчанию 32
- **inSharedDriveList** – Перечень общих дисков, доступ к которым предоставить на сторону удаленной RDP сессии. Пример: [«c», «d»]
- **inRedirectClipboardBool** – True - Синхронизировать буфер обмена между сессией Оркестратора и удаленной RDP сессией; False - Не синхронизировать буфер обмена. По

умолчанию True (в целях обратной совместимости). !ВНИМАНИЕ! Для учетных записей роботов мы рекомендуем не синхронизировать буфер обмена, так как это может привести к ошибкам роботов, которые работают с клавиатурой и буфером обмена

Результат:

```
{
```

«Host»: inHostStr, # Адрес хоста, пример «77.77.22.22» «Port»: str(inPortInt), # RDP порт, пример «3389» «Login»: inLoginStr, # Логин УЗ, пример «test» «Password»: inPasswordStr, # Пароль УЗ, пример «test» «Screen»: {

»Width»: inWidthPXInt, # Разрешение ширины RDP графической сессии в пикселях. По умолчанию 1680 «Height»: inHeightPXInt, Разрешение высоты RDP графической сессии в пикселях. По умолчанию 1050 «FlagUseAllMonitors»: inUseBothMonitorBool, «DepthBit»: str(inDepthBitInt)

}, «SharedDriveList»: inSharedDriveList, «RedirectClipboardBool»: True, ###### PROGRAM VARIABLE ############ «SessionHex»: «77777sdfsdf77777dsfdfsf77777777», «SessionIsWindowExistBool»: False, «SessionIsWindowResponsibleBool»: False, «SessionIsIgnoredBool»: False

}

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

L+,W+: Добавить активность по расписанию. Допускается указание времени и дней недели, в которые производить запуск.

```
# ПРИМЕР
from pyOpenRPA import Orchestrator

# ВАРИАНТ 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)
```

Параметры:

- **inGSettings** – Глобальный словарь настроек Оркестратора (синглтон)
- **inTimeHHMMStr** – Время запуска активности. Допускаются значения от «00:00» до «23:59». Example: «05:29»
- **inWeekdayList** – Список (list) дней недели, в которые выполнять запуск список активностей (ActivityItem list). 0 (понедельник) - 6 (воскресенье). Пример: [0,1,2,3,4]. По умолчанию устанавливается каждый день [0,1,2,3,4,5,6]
- **inActivityList** – Список активностей (ActivityItem list) на исполнение

pyOpenRPA.Orchestrator.__Orchestrator__.StorageRobotExists(*inRobotNameStr*) [исходный код]

L+,W+: Проверить, существует ли ключ inRobotNameStr в хранилище пользовательской информации StorageDict (GSettings > StarageDict)

Параметры:

inRobotNameStr – Наименование (ключ) робота. !ВНИМАНИЕ! Наименование чувствительно к регистру

Результат:

True - ключ робота присутствует в хранилище; False - отсутствует

pyOpenRPA.Orchestrator.__Orchestrator__.StorageRobotGet(*inRobotNameStr*) [исходный код]

L+,W+: Получить содержимое по ключу робота inRobotNameStr в хранилище пользовательской информации StorageDict (GSettings > StarageDict)

Параметры:

inRobotNameStr – Наименование (ключ) робота. !ВНИМАНИЕ! Наименование чувствительно к регистру

Результат:

Значение или структура, которая расположена по адресу (GSettings > StarageDict > inRobotNameStr)

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

L+,W+: [ПРЕКРАЩЕНИЕ ПОДДЕРЖКИ В 1.3.2, см. WebUserUACCheck] Проверить права доступа для пользователя запроса по списку ключей до права.

```
# ВАРИАНТ ИСПОЛЬЗОВАНИЯ 1 (инициализация модуля ру без вызова каких-либо функций внутри)
# автоинициализация всех .ру файлов, с префиксом CP_, которые расположены в папке ControlPanel
Orchestrator.UACKeyListCheck(inRequest=lRequest, inRoleKeyList=["ROBOT1","DISPLAY_DASHBOARD"])
```

Параметры:

- **inRequest** – Экземпляр request (from http.server import BaseHTTPRequestHandler)
- **inRoleKeyList** – Список ключей, права доступа к которому требуется проверить

Результат:

True - Пользователь обладает соответствующим правом; False - Пользователь не обладает соответствующим правом

pyOpenRPA.Orchestrator.__Orchestrator__.UACSuperTokenUpdate(*inSuperTokenStr*, *inGSettings=None*) [исходный код]

L+,W+: Добавить супертокен (полный доступ). Супертокены используются для подключения к Оркестратору по API

Параметры:

- **inGSettings** – Глобальный словарь настроек Оркестратора (синглтон)
- **inSuperTokenStr** – Кодовая строковая комбинация, которая будет предоставлять доступ роботу / агенту для взаимодействия с Оркестратором по API

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

L+,W+: Дообогащение словаря доступа UAC пользователя inADStrinADLoginStr. Если ранее словарь не устанавливался, то по умолчанию он {}. Далее идет дообогащение теми ключами, которые присутствуют в inRoleHierarchyAllowedDict

Параметры:

- **inGSettings** – Глобальный словарь настроек Оркестратора (синглтон)
- **inADLoginStr** – Логин пользователя
- **inADStr** – Домен пользователя. Если пусто - локальный компьютер или домен по-

умолчанию, который настроен в ОС

- **inADIsDefaultBool** – True - домен настроен по умолчанию; False - домен требуется обязательно указывать
- **inURLList** –
  Список разрешенных URL для пользователя. Для добавления URL рекомендуем использовать функции WebURLConnectDef, WebURLConnectFile, WebURLConnectFolder
  Формат: {
  »Method»: «GET» || «POST», «URL»: «/GetScreenshot», «MatchType»: «BeginWith» || «Equal» || «EqualCase» || «Contains» || «EqualNoParam», «ResponseDefRequestGlobal»: Функция python || «ResponseFilePath»: Путь к файлу || «ResponseFolderPath»: Путь к папке, в которой искать файлы, «ResponseContentType»: пример MIME «image/png», «UACBool»:False - не выполнять проверку прав доступа по запросу, «UseCacheBool»: True - кэшировать ответ},
- **inRoleHierarchyAllowedDict** – Словарь доступа пользователя UAC. Пустой словарь - полный доступ

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

L+,W+: [ПРЕКРАЩЕНИЕ ПОДДЕРЖКИ В 1.3.2, см. WebUserUACHierarchyGet] Вернуть UAC (User Access Control) словарь доступов для пользователя, который отправил запрос. Пустой словарь - супердоступ (доступ ко всему)

Параметры:

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

Результат:

Словарь доступов пользователя. Пустой словарь - супердоступ (доступ ко всему)

pyOpenRPA.Orchestrator.__Orchestrator__.WebAppGet()→ FastAPI [исходный код]

L+,W+: Вернуть экземпляр веб сервера fastapi.FastAPI (app). Подробнее про app см. https://fastapi.tiangolo.com/tutorial/first-steps/

pyOpenRPA.Orchestrator.__Orchestrator__.WebAuditMessageCreate(*inAuthTokenStr: Optional[str] = None*, *inHostStr: Optional[str] = None*, *inOperationCodeStr: str = '-'*, *inMessageStr: str = '-'*) [исходный код]

L+,W+: [ИЗМЕНЕНИЕ В 1.3.1] Создание сообщения ИТ аудита с такими сведениями как (Домен, IP, логин и тд.). Данная функция особенно актуальна в том случае, если требуется реализовать дополнительные меры контроля ИТ системы.

```
# ПРИМЕР
from pyOpenRPA import Orchestrator

lWebAuditMessageStr = Orchestrator.WebAuditMessageCreate(
    inRequest = lRequest,
    inOperationCodeStr = "OP_CODE_1",
    inMessageStr="Success"):

# Логгирование сообщения
lLogger.info(lWebAuditMessageStr)
```

Параметры:

- **inAuthTokenStr** (*str*, *опционально*) – Токен авторизации пользователя / бота, по умолчанию None (не установлен)
- **inHostStr** (*str*, *опционально*) – IP адрес хоста пользователя / бота, по умолчанию None (не установлен)
- **inOperationCodeStr** – Код операции, который принят в компании в соответствии с регламентными процедурами
- **inMessageStr** – Дополнительное сообщение, которое необходимо отправить в сообщение

об ИТ аудите

Результат:
Формат сообщения: «WebAudit :: DOMAINUSER@101.121.123.12 :: operation code :: message»

pyOpenRPA.Orchestrator.__Orchestrator__.WebAuthDefGet() [исходный код]

Вернуть функцию авторизации пользователя. Функция может использоваться для доменной авторизации.

```
# ПРИМЕР Если требуется авторизация пользователя (получить inAuthTokenStr) from fastapi import Request from fastapi.responses import JSONResponse, HTMLResponse from pyOpenRPA import Orchestrator @app.post(«/url/to/def»,response_class=JSONResponse) async def some_def(inRequest:Request, inAuthTokenStr:str=Depends(Orchestrator.WebAuthDefGet())):
    l_input_dict = await inRequest.json() if lValueStr == None or lValueStr == b»»: lValueStr=»» else: lValueStr = lValueStr.decode(«utf8»)
```

Результат:
Функция авторизации

Тип результата:
def

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

L+,W+: Добавить панель управления робота в Оркестратор. Для панели управления открыт доступ для использования функции-генератора HTML, генератора JSON данных, генератора JS кода.

Параметры:

- **inGSettings** – Глобальный словарь настроек Оркестратора (синглтон)
- **inCPKeyStr** – Текстовый ключ панели управления. Ключ для каждой панели управления должен быть уникальным!
- **inHTMLRenderDef** – Функция Python для генерации HTML кода. Для использования Jinja2 шаблонов HTML см. pyOpenRPA.Orchestrator.Managers.ControlPanel
- **inJSONGeneratorDef** – Функция Python для генерации JSON контейнера для отправки на клиентскую часть Оркестратора
- **inJSInitGeneratorDef** – Функция Python для генерации JS кода для отправки на клиентскую часть Оркестратора

pyOpenRPA.Orchestrator.__Orchestrator__.WebListenCreate(*inServerKeyStr='Default', inAddressStr='0.0.0.0', inPortInt=1024, inCertFilePEMPathStr=None, inKeyFilePathStr=None, inGSettings=None*) [исходный код]

L+,W+: Настроить веб-сервер Оркестратора.

Параметры:

- **inAddressStr** – IP адрес для прослушивания. Если «0.0.0.0», то прослушивать запросы со всех сетевых карт. Если «127.0.0.1», то слушать запросы только с той ОС, на которой работает Оркестратор
- **inPortInt** – Номер порта для прослушивания. Если HTTP - 80; Если HTTPS - 443. По умолчанию 1024 (Связано с тем, что в линукс можно устанавливать порты выше 1000). Допускается установка других портов
- **inCertFilePEMPathStr** – Путь файлу сертификата, сгенерированного в .pem (base64) формате. Обязателен при использовании защищенного HTTPS/SSL соединения.

- **inKeyFilePathStr** – Путь к файлу закрытого ключа в base64 формате
- **inGSettings** – Глобальный словарь настроек Оркестратора (синглтон)

Результат:

pyOpenRPA.Orchestrator.__Orchestrator__.WebRequestGet() [исходный код]

L+,W+: [ПРЕКРАЩЕНИЕ ПОДДЕРЖКИ В 1.3.2] Вернуть экземпляр HTTP запроса, если функция вызвана в потоке, который был порожден для отработки HTTP запроса пользователя.

pyOpenRPA.Orchestrator.__Orchestrator__.WebRequestHostGet(*inRequest*)→ str [исходный код]

L+,W+: Получить наименование хоста, с которого поступил запрос

Параметры:
inRequest (*fastapi.Request*, *опционально*) – Экземпляр fastapi.Request, по умолчанию None

Результат:
Наименование хоста

Тип результата:
str

pyOpenRPA.Orchestrator.__Orchestrator__.WebRequestParseBodyBytes(*inRequest=None*) [исходный код]

L+,W+: [ПРЕКРАЩЕНИЕ ПОДДЕРЖКИ В 1.3.1, см. FASTAPI] Извлечь данные в байт виде из тела (body) HTTP запроса.

Параметры:
inRequest – Экземпляр HTTP request. Опционален, если сообщение фиксируется из под потока, который был инициирован запросом пользователя

Результат:
Строка байт b““ или None (если тело запроса было пустым)

pyOpenRPA.Orchestrator.__Orchestrator__.WebRequestParseBodyJSON(*inRequest=None*) [исходный код]

L+,W+: [ПРЕКРАЩЕНИЕ ПОДДЕРЖКИ В 1.3.1, см. FASTAPI] Извлечь из тела (body) запроса HTTP JSON данные и преобразовать в Dict / List структуры языка Python.

Параметры:
inRequest – Экземпляр HTTP request. Опционален, если сообщение фиксируется из под потока, который был инициирован запросом пользователя

Результат:
Словарь (dict), список (list) или None (если тело запроса было пустым)

pyOpenRPA.Orchestrator.__Orchestrator__.WebRequestParseBodyStr(*inRequest=None*) [исходный код]

L+,W+: [ПРЕКРАЩЕНИЕ ПОДДЕРЖКИ В 1.3.1, см. FASTAPI] Извлечь данные в виде строки из тела (body) HTTP запроса.

Параметры:
inRequest – Экземпляр HTTP request. Опционален, если сообщение фиксируется из под потока, который был инициирован запросом пользователя

Результат:
Текстовая строка „“ или None (если тело запроса было пустым)

pyOpenRPA.Orchestrator.__Orchestrator__.WebRequestParseFile(*inRequest=None*) [исходный код]

L+,W+: [ПРЕКРАЩЕНИЕ ПОДДЕРЖКИ В 1.3.1, см. FASTAPI] Извлечь файл (наименование + содержимое в виде строки байт b““) из HTTP запроса пользователя.

Параметры:

inRequest – Экземпляр HTTP request. Опционален, если сообщение фиксируется из под потока, который был инициирован запросом пользователя

Результат:

Кортеж формата (FileNameStr, FileBodyBytes) or (None, None), если файл не был обнаружен

---

pyOpenRPA.Orchestrator.__Orchestrator__.WebRequestParsePath(*inRequest=None*) [исходный код]

L+,W+: [ПРЕКРАЩЕНИЕ ПОДДЕРЖКИ В 1.3.1, см. FASTAPI] Извлечь декодированный URL путь из HTTP запроса пользователя в формате строки.

Параметры:

inRequest – Экземпляр HTTP request. Опционален, если сообщение фиксируется из под потока, который был инициирован запросом пользователя

Результат:

str, пример: /pyOpenRPA/Debugging/DefHelper

---

pyOpenRPA.Orchestrator.__Orchestrator__.WebRequestResponseSend(*inResponeStr, inRequest=None, inContentTypeStr: Optional[str] = None, inHeadersDict: Optional[dict] = None*) [исходный код]

L+,W+: [ПРЕКРАЩЕНИЕ ПОДДЕРЖКИ В 1.3.1, см. FASTAPI] Установить ответ на HTTP запрос пользователя.

Параметры:

- **inResponeStr** – Тело ответа (строка)
- **inRequest** – Экземпляр HTTP request. Опционален, если сообщение фиксируется из под потока, который был инициирован запросом пользователя
- **inContentTypeStr** – МИМЕ тип. Пример: «html/text»
- **inHeadersDict** – Словарь (dict) ключей, которые добавить в headers HTTP ответа на запрос пользователя

---

pyOpenRPA.Orchestrator.__Orchestrator__.WebURLConnectDef(*inMethodStr, inURLStr, inMatchTypeStr, inDef, inContentTypeStr='application/octet-stream', inGSettings=None, inUACBool=None*) [исходный код]

L+,W+: Подключить функцию Python к URL.

Параметры:

- **inMethodStr** – Метод доступа по URL «GET» || «POST»
- **inURLStr** – URL адрес. Пример «/index»
- **inMatchTypeStr** – Тип соответсвия строки URL с inURLStr: «BeginWith» || «Contains» || «Equal» || «EqualCase» || «EqualNoParam»
- **inDef** – Функция Python. Допускаются функции, которые принимают следующие наборы параметров: 2:[inRequest, inGSettings], 1: [inRequest], 0: []
- **inContentTypeStr** – МИМЕ тип. По умолчанию: «application/octet-stream»
- **inUACBool** – True - Выполнять проверку прав доступа пользователя перед отправкой ответа; False - не выполнять проверку прав доступа пользователя
- **inGSettings** – Глобальный словарь настроек Оркестратора (синглтон)

---

pyOpenRPA.Orchestrator.__Orchestrator__.WebURLConnectFile(*inMethodStr, inURLStr, inMatchTypeStr, inFilePathStr, inContentTypeStr=None, inGSettings=None, inUACBool=None, inUseCacheBool=False*) [исходный код]

L+,W+: Подключить файл к URL.

Параметры:

- **inMethodStr** – Метод доступа по URL «GET» || «POST»

- **inURLStr** – URL адрес. Пример «/index»
- **inMatchTypeStr** – Тип соответсвия строки URL с inURLStr: «BeginWith» || «Contains» || «Equal» || «EqualCase» || «EqualNoParam»
- **inFilePathStr** – Путь к файлу на диске, который возвращать пользователю по HTTP
- **inContentTypeStr** – MIME тип. Если None - выполнить автоматическое определение
- **inUACBool** – True - Выполнять проверку прав доступа пользователя перед отправкой ответа; False - не выполнять проверку прав доступа пользователя
- **inUseCacheBool** – True - выполнить кэширование страницы, чтобы в следующих запросах открыть быстрее; False - не кэшировать
- **inGSettings** – Глобальный словарь настроек Оркестратора (синглтон)

pyOpenRPA.Orchestrator.__Orchestrator__.WebURLConnectFolder(*inMethodStr*, *inURLStr*, *inMatchTypeStr*, *inFolderPathStr*, *inExceptionFlagBool=False*, *inGSettings=None*, *inUACBool=None*, *inUseCacheBool=False*) [исходный код]

L+,W+: Подключить папку к URL.

Параметры:

- **inMethodStr** – Метод доступа по URL «GET» || «POST»
- **inURLStr** – URL адрес. Пример «/index»
- **inMatchTypeStr** – Тип соответсвия строки URL с inURLStr: «BeginWith» || «Contains» || «Equal» || «EqualCase» || «EqualNoParam»
- **inFolderPathStr** – Путь к папке на диске, в которой искать файл и возвращать пользователю по HTTP
- **inExceptionFlagBool** – Флаг на обработку ошибки. True - показывать ошибку в терминале (остановка инициализации), False - не показывать
- **inUACBool** – True - Выполнять проверку прав доступа пользователя перед отправкой ответа; False - не выполнять проверку прав доступа пользователя
- **inUseCacheBool** – True - выполнить кэширование страницы, чтобы в следующих запросах открыть быстрее; False - не кэшировать
- **inGSettings** – Глобальный словарь настроек Оркестратора (синглтон)

pyOpenRPA.Orchestrator.__Orchestrator__.WebURLIndexChange(*inURLIndexStr: str = '/'*) [исходный код]

L+,W+: Изменить адрес главной страницы Оркестратора. По умолчанию „/“

Параметры:

inURLIndexStr (*str*, *опционально*) – Новый адрес главной страницы Оркестратора.

pyOpenRPA.Orchestrator.__Orchestrator__.WebUserDomainGet(*inAuthTokenStr: Optional[str] = None*)→ str [исходный код]

L+,W+: Получить домен авторизованного пользователя. Если авторизация не производилась - вернуть None

Параметры:

inAuthTokenStr (*str*, *опционально*) – Токен авторизации пользователя / бота, по умолчанию None (не установлен)

Результат:

Домен пользователя

Тип результата:

str

pyOpenRPA.Orchestrator.__Orchestrator__.WebUserInfoGet(*inAuthTokenStr=None*) [исходный код]

L+,W+: Информация о пользователе, который отправил HTTP запрос.

Параметры:

inRequest – Экземпляр HTTP request. Опционален, если сообщение фиксируется из под потока, который был инициирован запросом пользователя

Результат:
Сведения в формате {«DomainUpperStr»: «PYOPENRPA», «UserNameUpperStr»: «IVAN.MASLOV»}

---

pyOpenRPA.Orchestrator.__Orchestrator__.WebUserIsSuperToken(*inAuthTokenStr: Optional[str] = None*)
[исходный код]

L+,W+: [ИЗМЕНЕНИЕ В 1.3.1] Проверить, авторизован ли HTTP запрос с помощью супер токена (токен, который не истекает).

Параметры:
inAuthTokenStr (*str*, *опционально*) – Токен авторизации пользователя / бота, по умолчанию None (не установлен)

Результат:
True - является супертокеном; False - не является супертокеном; None - авторизация не производилась

---

pyOpenRPA.Orchestrator.__Orchestrator__.WebUserLoginGet(*inAuthTokenStr: Optional[str] = None*)→ str
[исходный код]

L+,W+: Получить логин авторизованного пользователя. Если авторизация не производилась - вернуть None

Параметры:
inAuthTokenStr (*str*, *опционально*) – Токен авторизации пользователя / бота, по умолчанию None (не установлен)

Результат:
Логин пользователя

Тип результата:
str

---

pyOpenRPA.Orchestrator.__Orchestrator__.WebUserUACCheck(*inAuthTokenStr: Optional[str] = None*, *inKeyList: Optional[list] = None*)→ bool [исходный код]

L+,W+: Проверить UAC доступ списка ключей для пользователя

Параметры:
inAuthTokenStr (*str*, *опционально*) – Токен авторизации пользователя / бота, по умолчанию None (не установлен)

Результат:
True - доступ имеется, False - доступа нет

Тип результата:
bool

---

pyOpenRPA.Orchestrator.__Orchestrator__.WebUserUACHierarchyGet(*inAuthTokenStr: Optional[str] = None*)→ dict
[исходный код]

L+,W+: [ИЗМЕНЕНИЕ В 1.3.1] Вернуть словарь доступа UAC в отношении пользователя, который выполнил HTTP запрос inRequest

Параметры:
inAuthTokenStr (*str*, *опционально*) – Токен авторизации пользователя / бота, по умолчанию None (не установлен)

Результат:

UAC словарь доступа или {}, что означает полный доступ

Быстрая навигация

- Сообщество pyOpenRPA (telegram)
- Сообщество pyOpenRPA (tenchat)
- Сообщество pyOpenRPA (вконтакте)
- Презентация pyOpenRPA
- Портал pyOpenRPA
- Репозиторий pyOpenRPA

.. v1.4.0 replace:: v1.4.0

Предыдущая Следующая

3. Настройки GSettings (шаблон)

Общее

Ниже представлена структура единого глобального словаря настроек GSettings

Структура

```
import os, logging, datetime, sys
import schedule

from pyOpenRPA.Tools import CrossOS # https://schedule.readthedocs.io/en/stable/examples.html

# 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 
            "IntervalSecFloat": 3600.0,  # Sec float to periodically clear gsettings
            "AgentActivityReturnLifetimeSecFloat": 300.0 # Time in seconds to life for activity result re
        },
        "Client": {  # Settings about client web orchestrator
            "Session": {
                # Settings about web session. Session algorythms works only for special requests (URL in 
                "LifetimeSecFloat": 600.0,
                # Client Session lifetime in seconds. after this time server will forget about this clien
                "LifetimeRequestSecFloat": 120.0,  # 1 client request lifetime in server in seconds
                "ControlPanelRefreshIntervalSecFloat": 2.0,  # Interval to refresh control panels for ses
                "TechnicalSessionGUIDCache": {  # TEchnical cache. Fills when web browser is requesting
                    # "SessionGUIDStr":{ # Session with some GUID str. On client session guid stored in c
                    #     "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 j
                    #         }
                    #     },
                    #     "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 exist
                    # }
                },
            },
            # # # # # # 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": {
            "ControlPanelDict": {
                # "CPKey": <Managers.ControlPanel instance>
            },
            "URLIndexStr":"/", # The index html page for the orchestrator control panel. Default is /
            "AgentLimitLogSizeBytesInt": 300, # Don't show body if json body of transmition is more than
            "ServerThread": None, # Server thread is there
            "AgentActivityLifetimeSecFloat": 1200.0,  # Time in seconds to life for activity for the agen
            "AgentConnectionLifetimeSecFloat": 300.0,  # Time in seconds to handle the open connection to
            "AgentLoopSleepSecFloat": 2.0,  # Time in seconds to sleep between loops when check to send s
            "AgentFileChunkBytesSizeInt": 50000000, # size of the each chunk for the agent transmition
            "AgentFileChunkCheckIntervalSecFloat": 0.2, # The interval for check last activity item was s
            "WorkingDirectoryPathStr": None,  # Will be filled automatically
            "RequestTimeoutSecFloat": 300,  # Time to handle request in seconds,
            "ListenDict": { # Prototype
```

```
            #"Default":{
            #    "AddressStr":"",
            #    "PortInt":80,
            #    "CertFilePEMPathStr":None,
            #    "KeyFilePathStr":None,
            #    "ServerInstance": None
            #}
        },
        "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 virtu
                #               "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
            }
        },
        "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
            #    "UACBool": True # True - check user access before do this URL item. None - get Serve
            # }
            #{
            #    "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
            #    # "UACBool": True # True - check user access before do this URL item
            #}
        ],

    },
    "OrchestratorStart": {
        "DefSettingsUpdatePathList": [],
        # List of the .py files which should be loaded before init the algorythms
        "ActivityList": []
    },
    "SchedulerDict": {
        "Schedule": schedule, # https://schedule.readthedocs.io/en/stable/examples.html
        "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 ap
                #    "ActivityList": [
                #                    # {
                #                #    "Def":"DefAliasTest", # def link or def alias (look gSettings["Pro
                #                #    "ArgList":[1,2,3], # Args list
                #                #    "ArgDict":{"ttt":1,"222":2,"dsd":3} # Args dictionary
                #                #    "ArgGSettings": # Name of GSettings attribute: str (ArgDict) or in
                #                #    "ArgLogger": None # Name of GSettings attribute: str (ArgDict) or
                #                #    "GUIDStr": ..., # GUID of the activity
                #                # },
                #    ],
                #    "GUID": None # Will be filled in Orchestrator automatically - is needed for detect a
                # },
            ],
        },
        "ManagersProcessDict":{}, # The key of the Process is (mAgentHostNameStr.upper(), mAgentUserNameS
        "ManagersGitDict":{}, # The key of the Git instance is (mAgentHostNameStr.upper(), mAgentUserName.
        "ProcessorDict": {  # Has been changed. New general processor (one threaded) v.1.2.0
            "ActivityList": [  # List of the activities
                # {
                #    "Def":"DefAliasTest", # def link or def alias (look gSettings["Processor"]["AliasDefi
                #    "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
                # },
            ],
            "ActivityItemNowDict": None, # Activity Item which is executing now
            "AliasDefDict": {},  # Storage for def with Str alias. To use it see pyOpenRPA.Orchestrator.C
            "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
        },
        # # # # # # # # # # # # # #
        "RobotRDPActive": {
            "RecoveryDict": {
                "CatchPeriodSecFloat": 1200, # Catch last 10 minutes
                "TriggerCountInt": 10, # Activate trigger if for the period orch will catch the reconnect
                "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 sc
                #        "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 "Flag
                #    "SessionIsWindowResponsibleBool": False, # Flag if RDP window is responsible (recieve
                #    "SessionIsIgnoredBool": False # Flag to ignore RDP window False - dont ignore, True
                # }
            },
            "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 fulls
            "ActivityList": [
                # Technical Activity list for RobotRDPActive thread - equal to Main activity list, apply
                # {
                #    "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"
                #                 "inLoginStr": "login", "inPasswordStr": "pass"}  # Args dictionary
                # },
                # {
                #     "DefNameStr": "RDPSessionDisconnect",  # Disconnect the RDP session without logoff.
                #     "ArgList": [],  # Args list
                #     "ArgDict": {"inRDPSessionKeyStr": "TestRDP"}
                # },
                # {
                #     "DefNameStr": "RDPSessionReconnect",  # Disconnect the RDP session without logoff. F
                #     "ArgList": [],  # Args list
                #     "ArgDict": {"inRDPSessionKeyStr": "TestRDP"}
                # }
            ]
        },
        # # # # # # # # # # # # # #
        "FileManager": {
            "FileURLFilePathDict_help": "https://localhost:8081/filemanager/<file URL>. All FileURL s mus
            "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 exe
            # 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, "Activity

# 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 cont
                # }
            },
            "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 o
                "RestartPCBool": True, # Send CMD to restart pc
                "NothingBool":True, # USe option if you dont want to give some access to the RDP controls
                "Debugging":True # Debugging tool
            },
            "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", inLoggerLevel = None):
    if inModeStr=="BASIC":
        lResult = __Create__() # Create settings
        # Создать файл логирования
        # add filemode="w" to overwrite
        if not os.path.exists("Logs"):
            os.makedirs("Logs")
        ##########################
        # Подготовка логгера Robot
        ##########################
        if inLoggerLevel is None: inLoggerLevel=logging.INFO
        lL = lResult["Logger"]
        if len(lL.handlers) == 0:
            lL.setLevel(logging.INFO)
            # create the logging file handler
            mRobotLoggerFH = logging.FileHandler(
                CrossOS.PathStr("Logs\\" + 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
            lL.addHandler(mRobotLoggerFH)
            ####################Add console output
            handler = logging.StreamHandler(sys.stdout)
            handler.setFormatter(mRobotLoggerFormatter)
            lL.addHandler(handler)
            ##########################################
            LoggerDumpLogHandlerAdd(inLogger=lL, inGSettingsClientDict=lResult["Client"])
            #mHandlerDumpLogList = LoggerHandlerDumpLogList.LoggerHandlerDumpLogList(inDict=lResult["Clie
            #                                                                         inKeyStr="DumpLogLis
            #                                                                         inHashKeyStr="DumpLo
            #                                                                         inRowCountInt=lResul
            #                                                                             "DumpLogListCoun
            #mHandlerDumpLogList.setFormatter(mRobotLoggerFormatter)
            #mRobotLogger.addHandler(mHandlerDumpLogList)
        else:
            if lL: lL.warning("Внимание! Вручную была вызвана функция SettingsTemplate.Create - начиная с
    return lResult # return the result dict
```

Быстрая навигация

- Сообщество pyOpenRPA (telegram)
- Сообщество pyOpenRPA (tenchat)
- Сообщество pyOpenRPA (вконтакте)
- Презентация pyOpenRPA
- Портал pyOpenRPA
- Репозиторий pyOpenRPA

.. v1.4.0 replace:: v1.4.0

Предыдущая Следующая

4. Как использовать?

Как запустить?

Хотите выполнить запуск Оркестратора?

Для этого достаточно (выбрать одно):

- запустить демо-стэнд: запустить .cmd файл, расположенный в папке pyOpenRPA по адресу: Orchestratorstart.cmd (для Windows) и start.sh (для Linux). Далее перейти в браузер по адресу: http://localhost:1024
- в свой .py скрипт добавить следующий код (см. ниже)

```
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 -
    # Call the orchestrator main def
    Orchestrator.Orchestrator(inGSettings=gSettings)
```

Шаблоны функций веб-сервера (с использованием FastAPI)

```
# ПРИМЕР Если НЕ требуется авторизация пользователя (получить inAuthTokenStr)
from fastapi import Request
from fastapi.responses import JSONResponse, HTMLResponse
@app.post("/url/to/def",response_class=JSONResponse)
async def some_def(inRequest:Request):
    l_input_dict = await inRequest.json()
    if lValueStr == None or  lValueStr == b"": lValueStr=""
    else: lValueStr = lValueStr.decode("utf8")

# ПРИМЕР Если требуется авторизация пользователя (получить inAuthTokenStr)
from fastapi import Request
from fastapi.responses import JSONResponse, HTMLResponse
from pyOpenRPA import Orchestrator
@app.post("/url/to/def",response_class=JSONResponse)
async def some_def(inRequest:Request, inAuthTokenStr:str=Depends(Orchestrator.WebAuthDefGet())):
    l_input_dict = await inRequest.json()
    if lValueStr == None or  lValueStr == b"": lValueStr=""
    else: lValueStr = lValueStr.decode("utf8")
```

Конфигурационный файл config.py

Также вы можете выполнить более тонкую настройку параметров Оркестратора. Ниже пример такой настройки:

```
import psutil, datetime, logging, os, sys


# Config settings
lPyOpenRPASourceFolderPathStr = (r"../Sources") # Path for test pyOpenRPA package

# Operations
if lPyOpenRPASourceFolderPathStr != "": sys.path.insert(0,os.path.abspath(os.path.join(lPyOpenRPASourceFo


# Start import after config the pyOpenRPA folder
from pyOpenRPA.Orchestrator import SettingsTemplate # Import functionallity
from pyOpenRPA.Tools import CrossOS
from pyOpenRPA import Orchestrator  # Import orchestrator main
from pyOpenRPA.Orchestrator.Server import app
import threading

from fastapi import Depends
from fastapi.responses import PlainTextResponse
from fastapi.responses import FileResponse




# Пример создания функции на сервере (FASTAPI) /test/threads
@app.get(path="/test/threads",tags=["Test"],response_class=PlainTextResponse)
def Threads():# inAuthDict:dict=Depends(IdentifyAuthorize)
#def Threads(inAuthDict:dict=Depends(IdentifyAuthorize)):# Используй, если требуется авторизация
    lThreadStr = ""
    for thread in threading.enumerate():
        lThreadStr+=f"ПОТОК: {thread.name}\n"
    #print(thread.name)
    return lThreadStr


#Run as administrator (ONLY FOR WINDOWS)
if not Orchestrator.OrchestratorIsAdmin() and CrossOS.IS_WINDOWS_BOOL:
    Orchestrator.OrchestratorRerunAsAdmin()
    print(f"Orchestrator will be run as administrator!")
else:
    gSettings = Orchestrator.GSettingsGet()
    #gSettings = SettingsTemplate.Create(inModeStr="BASIC") # Create GSettings with basic configuration -
    Orchestrator.OrchestratorLoggerGet().setLevel(logging.INFO)
    # TEST Add User ND - Add Login ND to superuser of the Orchestrator
    lUACClientDict = SettingsTemplate.__UACClientAdminCreate__()
    gSettings["ServerDict"]["AccessUsers"]["FlagCredentialsAsk"]=True
    Orchestrator.UACUpdate(inGSettings=gSettings, inADLoginStr="ND", inADStr="", inADIsDefaultBool=True, 
    Orchestrator.UACUpdate(inGSettings=gSettings, inADLoginStr="rpa00", inADStr="", inADIsDefaultBool=Tru
    # TEST Add User IMaslov - Add Login IMaslov to superuser of the Orchestrator
    Orchestrator.UACUpdate(inGSettings=gSettings, inADLoginStr="VLADICK", inADStr="", inADIsDefaultBool=T
    # TEST Add Supertoken for the all access between robots
    Orchestrator.UACSuperTokenUpdate(inGSettings=gSettings, inSuperTokenStr="1992-04-03-0643-ru-b4ff-open
    # Add first interface!
    if CrossOS.IS_WINDOWS_BOOL:
        Orchestrator.WebListenCreate(inGSettings=gSettings, inPortInt=1024)
    if CrossOS.IS_LINUX_BOOL:
        Orchestrator.WebListenCreate(inGSettings=gSettings, inPortInt=1024)
    # Restore DUMP
    Orchestrator.OrchestratorSessionRestore(inGSettings=gSettings)
    # Autoinit control panels starts with CP_
    lPyModules = Orchestrator.OrchestratorPySearchInit(inGlobPatternStr="Demo\\*\\config.py", inAsyncInit
    #lPyModules2 = Orchestrator.OrchestratorPySearchInit(inGlobPatternStr="..\\..\\KPI_Effect\\packages\\

        # Call the orchestrator def
    Orchestrator.Orchestrator(inGSettings=gSettings, inDumpRestoreBool=False)
```

Быстрая навигация

- Сообщество pyOpenRPA (telegram)
- Сообщество pyOpenRPA (tenchat)
- Сообщество pyOpenRPA (вконтакте)

- Презентация pyOpenRPA
- Портал pyOpenRPA
- Репозиторий pyOpenRPA

.. v1.4.0 replace:: v1.4.0

Предыдущая Следующая

5. Права доступа пользователей UAC

Описание

Оркестратор обладает уникальным и гибким механизмом контроля и распределения прав доступа пользователей (UAC - User Access Control). Концепция pyOpenRPA заключается в том, что настройка доступа должна быть максимально функциональной, но при этом удобной в использовании.

Если требуется дать все права доступа - не нужно ничего редактировать - они будут по умолчанию.

Если нужно внести корректировки - вы можете указать лишь те ключи доступа, которые потребуются пользователю.

Также UAC механизм распределения прав доступа отлично интегрируется со сторонними системами безопасности, которые уже используются в компаниях, например Active Directory и другие.

Функции по работе с UAC представлены в группе `Orchestrator.UAC` в разделе описания функций: 2. Функции.

Если у вас останутся вопросы, то вы всегда можете обратиться в центр поддержки клиентов pyOpenRPA. Контакты: 2. Лицензия & Контакты

pyOpenRPA - роботы помогут!

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 r
        "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
    }
}
```

Быстрая навигация

- Сообщество pyOpenRPA (telegram)
- Сообщество pyOpenRPA (tenchat)
- Сообщество pyOpenRPA (вконтакте)
- Презентация pyOpenRPA
- Портал pyOpenRPA
- Репозиторий pyOpenRPA

.. v1.4.0 replace:: v1.4.0

Предыдущая Следующая