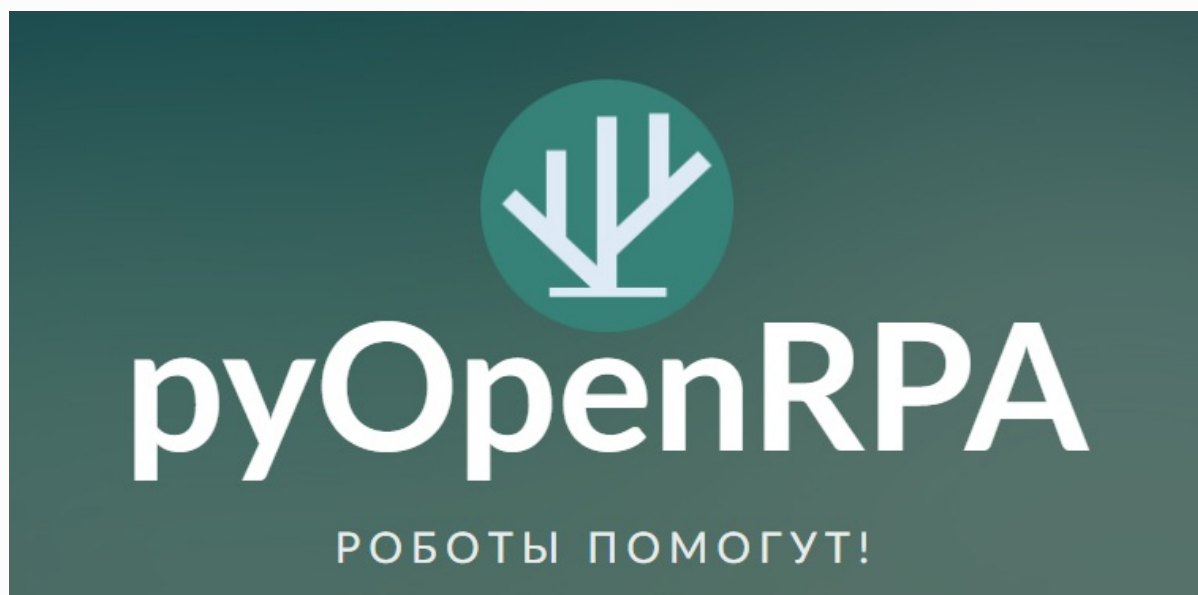


Портал документации ruOpenRPA



Дорогие коллеги!

Добро пожаловать на портал документации ruOpenRPA.

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

- 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 ruOpenRPA 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:

- **Вариант 1:** Подписка на премиальную техническую поддержку от вендора (ежемесячно от 66 900 руб. без НДС)
- **Вариант 2:** Покупка бессрочной лицензии на текущую версию ruOpenRPA и ниже (разовый платеж 299 000 руб. без НДС)
- **Вариант 3:** Покупка бессрочной лицензии на 1-го робота, который будет использовать текущую версию ruOpenRPA и ниже (разовый платеж 99 000 руб. без НДС)

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)

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

ВЕРСИЯ v1.2.13

! ВНИМАНИЕ ! Требуется пакет KB2999226 если используется windows Vista/7/8/8.1/Server 2008/Server 2012 <https://support.microsoft.com/ru-ru/help/2999226> Использование компонента OpenCV: ОС Windows 7/8/8/10 (Windows Server только с 2016) В июле 2022 будет выпущен релиз ruOpenRPA, который будет работать на ОС семейства Linux (Ubuntu / Astra)

- ОС Windows 7+ / Server 2008+.
- RAM 512+ Гб
- HDD 2+ Гб
- CPU 1+ 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 [\[\[ОТКРЫТЬ НАВР\]\]](#); [\[\[ОТКРЫТЬ GITLAB\]\]](#)
- RUS Практическое руководство по работе с Web UI [\[\[ОТКРЫТЬ НАВР\]\]](#); [\[\[ОТКРЫТЬ GITLAB\]\]](#)
- 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

- [Выбрать версию](#)
- [1. Первый запуск](#)
 - [Проверить, что ruOpenRPA развернута корректно?](#)
 - [Быстрая навигация](#)
- [2. Лицензия & Контакты](#)
 - [Лицензия](#)
 - [Автор](#)
 - [Правообладатель](#)
 - [Центр поддержки клиентов](#)
 - [Иван Маслов \(генеральный директор ООО «ОПЕН РПА»\)](#)
 - [Используемые сторонние компоненты \(лицензионная чистота\)](#)
 - [Быстрая навигация](#)

МОДУЛЬ РОБОТ

- [1. Описание](#)
 - [Общее](#)
 - [Примеры](#)
 - [Быстрая навигация](#)
- [2. Функции UIDesktop](#)
 - [Общее](#)
 - [Описание функций](#)
 - [Селектор UIO](#)
 - [Быстрая навигация](#)
- [3. Функции UIWeb](#)
 - [Общее](#)
 - [Быстрая навигация](#)

- 4. Функции Keyboard
 - Общее
 - Примеры использования
 - Описание функций
 - Коды клавиш
 - Дополнительная функциональность
 - Быстрая навигация
- 5. Функции Clipboard
 - Описание функций
 - Быстрая навигация
- 6. Функции Mouse
 - Общее
 - Описание функций
 - Быстрая навигация
- 7. Функции Screen
 - Общее
 - Класс Box
 - Класс Point
 - Символьное указание точки (inPointRuleStr)
 - Символьное указание области (inAnchorRuleStr)
 - Описание функций
 - Быстрая навигация
- 8. Как использовать?
 - Как запустить скрипт робота?
 - Быстрая навигация

МОДУЛЬ СТУДИЯ

- 1. Описание
 - Общее
 - Быстрая навигация
- 2. Как использовать?
 - Общее
 - Как запустить?
 - Описание UI студии
 - Извлечь UI дерево
 - Поиск UI объекта по наведению мыши
 - Извлечь свойства UI объекта
 - Быстрая навигация

МОДУЛЬ ОРКЕСТРАТОР

- 1. Описание
 - Общее
 - Концепция единого глобального словаря настроек (GSettings)
 - Архитектура
 - Быстрая навигация

- [2. Функции](#)
 - [Общее](#)
 - [Что такое активность \(ActivityItem\)?](#)
 - [Функции](#)
 - [Быстрая навигация](#)
- [3. Настройки GSettings \(шаблон\)](#)
 - [Общее](#)
 - [Структура](#)
 - [Быстрая навигация](#)
- [4. Как использовать?](#)
 - [Как запустить?](#)
 - [Параметры настройки](#)
 - [Быстрая навигация](#)
- [5. Права доступа пользователей UAC](#)
 - [Описание](#)
 - [UAC Dict for Orchestrator WEB UI rights](#)
 - [Быстрая навигация](#)

МОДУЛЬ АГЕНТ

- [2. Функции](#)
 - [pyOpenRPA.Agent.__Agent__](#)

ИНСТРУМЕНТЫ

- [2. Функции StopSafe](#)

Быстрая навигация

- [Сообщество pyOpenRPA \(telegram\)](#)
- [Сообщество pyOpenRPA \(tenchat\)](#)
- [Сообщество pyOpenRPA \(вконтакте\)](#)
- [Презентация pyOpenRPA](#)
- [Портал pyOpenRPA](#)
- [Репозиторий pyOpenRPA](#)

.. v1.2.13 replace:: v1.2.13

[Следующая](#) ➔

» 1. Первый запуск

1. Первый запуск

Готовы испытать всю мощь перспективных технологий?

Будет очень интересно - начинаем!

Для начала необходимо выполнить следующие действия:

- Скачать пакет pyOpenRPA с главной страницы <https://pyopenrpa.ru/>
- Распаковать пакет куда угодно!

ВСЁ - Развертывание pyOpenRPA завершено! :)

Проверить, что pyOpenRPA развернута корректно?

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).

```
OpenRPA\Resources\WPy64-3720\python-3.7.2.amd64\python.exe
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Быстрая навигация

- [Сообщество pyOpenRPA \(telegram\)](#)
- [Сообщество pyOpenRPA \(tenchat\)](#)
- [Сообщество pyOpenRPA \(вконтакте\)](#)
- [Презентация pyOpenRPA](#)
- [Портал pyOpenRPA](#)
- [Репозиторий pyOpenRPA](#)

.. v1.2.13 replace:: v1.2.13

Предыдущая

Следующая

2. Лицензия & Контакты

Лицензия

Лицензия pyOpenRPA разрешает бесплатное использование только для некоммерческих организаций и физических лиц (не ИП и не самозанятой). 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>
- Jupiter Notebook 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](#)

.. v1.2.13 replace:: v1.2.13

[← Предыдущая](#)

[Следующая →](#)

© Copyright 2022, ООО "ОПЕН РПА".

Собрано при помощи [Sphinx](#) с использованием [темы](#), предоставленной [Read the Docs](#).

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 экраном рабочего стола. Перейти к описанию функций: [7. Функции Screen](#)

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 представлен пример использования инструментов робота.

```
import time
from pyOpenRPA.Robot import UIDesktop
# UIDesktop: Работа с ОС
```

```

# Запустить задачу с UI
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", "b
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()

```

Быстрая навигация

- [Сообщество ruOpenRPA \(telegram\)](#)
- [Сообщество ruOpenRPA \(tenchat\)](#)
- [Сообщество ruOpenRPA \(вконтакте\)](#)
- [Презентация ruOpenRPA](#)
- [Портал ruOpenRPA](#)
- [Репозиторий ruOpenRPA](#)

.. v1.2.13 replace:: v1.2.13

[← Предыдущая](#)

[Следующая →](#)

© Copyright 2022, ООО "ОПЕН РПА".

Собрано при помощи [Sphinx](#) с использованием [темы](#), предоставленной [Read the Docs](#).

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

Functions:

<code>BackendStr_GetTopLevelList_UIOInfo</code> ((inBackend))	Получить список UIOInfo словарей - процессы, которые запущены в рабочей сессии и готовы для взаимодействия с роботом через backend inBackend
<code>Get_OSBitnessInt</code> ()	Определить разрядность робота, в котором запускается данная функция
<code>PWASpecification_Get_PWAApplication</code> (...)	Получить значение атрибута backend по PWA (pywinauto) селектору.
<code>PWASpecification_Get_UIO</code> (...)	Получить UIO объект по PWA (pywinauto) селектору.
<code>UIOEI_Convert_UIOInfo</code> (inElementInfo)	Техническая функция: Добогащение словаря с параметрами UIO объекта по заданному UIO.element_info
<code>UIOSelectorSecs_WaitAppear_Bool</code> (...)	Ожидать появление 1-го UIO объекта по заданному UIO селектору
<code>UIOSelectorSecs_WaitDisappear_Bool</code> (...)	Ожидать исчезновение 1-го UIO объекта по заданному UIO селектору
<code>UIOSelectorUIOActivity_Run_Dict</code> (...[, ...])	Выполнить активность inActionName над UIO объектом, полученным с помощью UIO селектора inUIOSelector.
<code>UIOSelector_Exist_Bool</code> (inUIOSelector)	Проверить существование хотя бы 1-го UIO объекта по заданному UIO селектору
<code>UIOSelector_FocusHighlight</code> (inUIOSelector)	Установить фокус и подсветить на несколько секунд на экране зеленой рамкой UIO объект, который соответствует входящему UIO селектору inUIOSelector
<code>UIOSelector_GetChildList_UIOList</code> ([...])	Получить список дочерних UIO объектов по входящему UIO селектору inUIOSelector.
<code>UIOSelector_Get_BitnessInt</code> (inSpecificationList)	Определить разрядность приложения по UIO селектору.
<code>UIOSelector_Get_BitnessStr</code> (inSpecificationList)	Определить разрядность приложения по UIO селектору.
<code>UIOSelector_get_UIO</code> (inSpecificationList[, ...])	Получить список UIO объект по UIO селектору.
<code>UIOSelector_Get_UIOActivityList</code> (inUIOSelector)	Получить список доступных действий/функций по UIO селектору inUIOSelector.
<code>UIOSelector_Get_UIOInfo</code> (inUIOSelector)	Получить свойства UIO объекта (element_info), по заданному UIO селектору.
<code>UIOSelector_Get_UIOInfoList</code> (inUIOSelector[, ...])	Техническая функция: Получить список параметров последних уровней UIO селектора по UIO объектам, которые удовлетворяют входящим inUIOSelector, поиск по которым будет производится от уровня inElement.
<code>UIOSelector_Get_UIOList</code> (inSpecificationList)	Получить список UIO объектов по UIO селектору
<code>UIOSelector_Highlight</code> (inUIOSelector)	Подсветить на несколько секунд на экране зеленой рамкой UIO объект, который соответствует входящему UIO селектору inUIOSelector
<code>UIOSelector_SafeOtherGet_Process</code> (inUIOSelector)	Получить процесс робота другой разрядности (если приложение UIO объекта выполняется в другой разрядности).
<code>UIOSelector_SearchChildByMouse_UIO</code> (...)	Инициировать визуальный поиск UIO объекта с помощью указателя мыши.

<code>UISelector_SearchChildByMouse_UIOTree (...)</code>	Получить список уровней UIO объекта с указанием всех имеющихся атрибутов по входящему UIO селектору.
<code>UISelector_SearchProcessNormalize_UIOSelector (...)</code>	Нормализовать UIO селектор для дальнейшего использования в функциях поиска процесса, в котором находится искомый UIO объект.
<code>UISelector_SearchUIONormalize_UIOSelector (...)</code>	Нормализовать UIO селектор для дальнейшего использования в функциях поиска UIO объекта.
<code>UISelector_TryRestore_Dict (inSpecificationList)</code>	Восстановить окно приложения на экране по UIO селектору inSpecificationList, если оно было свернуто.
<code>UISelectorsSecs_WaitAppear_List (...[, ...])</code>	Ожидать появления хотя бы 1-го / всех UIO объектов по заданным UIO селекторам
<code>UISelectorsSecs_WaitDisappear_List (...[, ...])</code>	Ожидать исчезновения хотя бы 1-го / всех UIO объектов по заданным UIO селекторам
<code>UIOXY_SearchChild_ListDict (inRootElement, ...)</code>	Техническая функция: Получить иерархию вложенности UIO объекта по заданным корневому UIO объекту, координатам X и Y.
<code>UIO_FocusHighlight (IWrapperObject[, colour, ...])</code>	Установить фокус и выполнить подсветку UIO объекта на экране
<code>UIO_GetCtrlIndex_Int (inElement)</code>	Получить индекс UIO объекта inElement в списке родительского UIO объекта.
<code>UIO_Highlight (IWrapperObject[, colour, ...])</code>	Выполнить подсветку UIO объекта на экране

`pyOpenRPA.Robot.UIDesktop.BackendStr_GetTopLevelList_UIOInfo(inBackend='win32')` [\[исходный код\]](#)

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.Get_OSBitnessInt()` [\[исходный код\]](#)

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)` [\[исходный код\]](#)

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
# IC: UIO Селектор выбора базы
lDemoBaseUIOSelector = [{"title": "Запуск IC: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 соответствия 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)` [\[исходный код\]](#)

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
IDemoBaseUIOSelector = [{"title": "Запуск 1C:Предприятия", "class_name": "V8TopLevelFrameTaxiStarter", "ba
lUIOObject = UIDesktop.PWASpecification_Get_UIO(IDemoBaseUIOSelector) # 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 соответствия 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.UIOEI_Convert_UIOInfo(inElementInfo)` [\[исходный код\]](#)

Техническая функция: Дообогачение 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
DemoBaseUIOSelector = [{"title": "Запуск 1C:Предприятия", "class_name": "V8TopLevelFrameTaxiStarter", "ba
lUIO = UIDesktop.UIOSelector_Get_UIO(1DemoBaseUIOSelector) # 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_Boo1(inSpecificationList, inWaitSecs)

[\[исходный код\]](#)

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 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 1C:Предприятия", "class_name": "V8TopLevelFrameTaxiStarter", "ba
lDemoBaseUIOExistBoo1 = UIDesktop.UIOSelectorSecs_WaitAppear_Boo1(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)

Результат:

True - UIO объект был обнаружен. False - обратная ситуация

pyOpenRPA.Robot.UIDesktop.UIOSelectorSecs_WaitDisappear_Boo1(inSpecificationList, inWaitSecs)

[\[исходный код\]](#)

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 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 1C:Предприятия", "class_name": "V8TopLevelFrameTaxiStarter", "ba
lDemoBaseUIOExistBoo1 = UIDesktop.UIOSelectorSecs_WaitDisappear_Boo1(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)

Результат:

True - UIO объект был обнаружен. False - обратная ситуация

pyOpenRPA.Robot.UIDesktop.UIOSelectorUIOActivity_Run_Dict(inUIOSelector, inActionName, inArgumentList=None, inkwArgumentObject=None) [\[исходный код\]](#)

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

СЕССИИ. 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
# 1С: UIO Селектор выбора базы
IDemoBaseUIOSelector = [{"title": "Запуск 1С:Предприятия", "class_name": "V8TopLevelFrameTaxiStarter", "ba
lActivityResult = UIDesktop.UIOSelectorUIOActivity_Run_Dict(IDemoBaseUIOSelector, "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
- **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) [\[исходный код\]](#)

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 СЕССИИ. 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
# 1С: UIO Селектор выбора базы
IDemoBaseUIOSelector = [{"title": "Запуск 1С:Предприятия", "class_name": "V8TopLevelFrameTaxiStarter", "ba
IDemoBaseUIOExistBool = UIDesktop.UIOSelector_Exist_Bool(IDemoBaseUIOSelector) # Получить булевый резу
```

Параметры:

inUIOSelector (*list, обязательный*) – UIO Селектор, который определяет критерии поиска UIO объектов

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) [\[исходный код\]](#)

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 СЕССИИ. 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
# 1С: UIO Селектор выбора базы
IDemoBaseUIOSelector = [{"title": "Запуск 1С:Предприятия", "class_name": "V8TopLevelFrameTaxiStarter", "ba
UIDesktop.UIOSelector_FocusHighlight(IDemoBaseUIOSelector) # 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') [\[исходный код\]](#)

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 СЕССИИ. 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
# 1С: UIО Селектор выбора базы
IDemoBaseUIOSelector = [{"title": "Запуск 1С:Предприятия", "class_name": "V8TopLevelFrameTaxiStarter", "ba
lUIOList = UIDesktop.UIOSelector_GetChildList_UIOList(IDemoBaseUIOSelector) # 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*) [\[исходный код\]](#)

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
# 1С: UIО Селектор выбора базы
IDemoBaseUIOSelector = [{"title": "Запуск 1С:Предприятия", "class_name": "V8TopLevelFrameTaxiStarter", "ba
IDemoBaseBitInt = UIDesktop.UIOSelector_Get_BitnessInt(IDemoBaseUIOSelector) # 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*) [\[исходный код\]](#)

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
# 1С: UIО Селектор выбора базы
IDemoBaseUIOSelector = [{"title": "Запуск 1С:Предприятия", "class_name": "V8TopLevelFrameTaxiStarter", "ba
IDemoBaseBitStr = UIDesktop.UIOSelector_Get_BitnessStr(IDemoBaseUIOSelector) # 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*) [\[исходный код\]](#)

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
# 1С: UIО Селектор выбора базы
IDemoBaseUIOSelector = [{"title": "Запуск 1С:Предприятия", "class_name": "V8TopLevelFrameTaxiStarter", "ba
IDemoBaseUIOList = UIDesktop.UIOSelector_Get_UIO(IDemoBaseUIOSelector) #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 обнаружена 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)` [\[исходный код\]](#)

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 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
# 1С: 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.

`pyOpenRPA.Robot.UIDesktop.UIOSelector_Get_UIOInfo(inUIOSelector)` [\[исходный код\]](#)

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 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
# 1С: UIO Селектор выбора базы
lDemoBaseUIOSelector = [{"title": "Запуск 1С:Предприятия", "class_name": "V8TopLevelFrameTaxiStarter", "ba
lUIOElementInfoDict = UIDesktop.UIOSelector_Get_UIOInfo(lDemoBaseUIOSelector) # Получить свойства над U
```

Параметры:

inUISelector (*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(inUISelector, inElement=None) [\[исходный код\]](#)

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

!ВНИМАНИЕ! ДАННАЯ ФУНКЦИОНАЛЬНОСТЬ В АВТОМАТИЧЕСКОМ РЕЖИМЕ ПОДДЕРЖИВАЕТ ВСЕ РАЗРЯДНОСТИ ПРИЛОЖЕНИЙ (32|64), КОТОРЫЕ ЗАПУЩЕНЫ В СЕССИИ. 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
# 1С: UIO Селектор выбора базы
DemoBaseUIOSelector = [{"title": "Запуск 1С:Предприятия", "class_name": "V8TopLevelFrameTaxiStarter", "ba
UIOInfoList = UIDesktop.UIOSelector_Get_UIOInfoList(DemoBaseUIOSelector) # Получить словарь параметр
```

Параметры:

- **inUISelector** (*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 inUISelector. По умолчанию None - поиск среди всех приложений.

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) [\[исходный код\]](#)

Получить список UIO объектов по UIO селектору

```
# UIDesktop: Взаимодействие с UI объектами приложений
from pyOpenRPA.Robot import UIDesktop
# 1С: UIO Селектор выбора базы
DemoBaseUIOSelector = [{"title": "Запуск 1С:Предприятия", "class_name": "V8TopLevelFrameTaxiStarter", "ba
DemoBaseUIOList = UIDesktop.UIOSelector_Get_UIOList(DemoBaseUIOSelector) #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 обнаружена ни одного UIO объекта по заданному UIO селектору. False - обратный случай. По умолчанию True

Результат:

Список UIO объектов, которые удовлетворяют условиям UIO селектора

pyOpenRPA.Robot.UIDesktop.UIOSelector_HighLight(inUISelector) [\[исходный код\]](#)

Подсветить на несколько секунд на экране зеленой рамкой UIO объект, который соответствует входящему UIO селектору inUISelector

!ВНИМАНИЕ! ДАННАЯ ФУНКЦИОНАЛЬНОСТЬ В АВТОМАТИЧЕСКОМ РЕЖИМЕ ПОДДЕРЖИВАЕТ ВСЕ РАЗРЯДНОСТИ ПРИЛОЖЕНИЙ (32|64), КОТОРЫЕ ЗАПУЩЕНЫ В СЕССИИ. 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 1C: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_SafeOtherGet_Process(*inUIOSelector*) [\[исходный код\]](#)

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 1C:Предприятия", "class_name": "V8TopLevelFrameTaxiStarter", "ba
lOtherBitnessProcess = UIDesktop.UIOSelector_SafeOtherGet_Process(lDemoBaseUIOSelector) # Вернуть про
```

Параметры:

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

Результат:

Процесс робота схожей разрядности

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

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 1C: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*) [\[исходный код\]](#)

Получить список уровней UIO объекта с указанием всех имеющихся атрибутов по входящему UIO селектору.

ВНИМАНИЕ! ДАННАЯ ФУНКЦИОНАЛЬНОСТЬ В АВТОМАТИЧЕСКОМ РЕЖИМЕ ПОДДЕРЖИВАЕТ ВСЕ РАЗРЯДНОСТИ ПРИЛОЖЕНИЙ (32|64), КОТОРЫЕ ЗАПУЩЕНЫ В СЕССИИ. 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 1C: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.

Результат:

list, список атрибутов на каждом уровне UIO объекта

pyOpenRPA.Robot.UIDesktop.UIOSelector_SearchProcessNormalize_UIOSelector(inControlSpecificationArray) [\[исходный код\]](#)

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
# 1С: UIO Селектор выбора базы
IDemoBaseUIOSelectorDirty = [{"title": "Запуск 1С:Предприятия", "class_name": "V8TopLevelFrameTaxiStarter"}]
IDemoBaseUIOSelectorClean = UIDesktop.UIOSelector_SearchProcessNormalize_UIOSelector(IDemoBaseUIOSelectorDirty)
```

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) [\[исходный код\]](#)

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
# 1С: UIO Селектор выбора базы
IDemoBaseUIOSelectorDirty = [{"title": "Запуск 1С:Предприятия", "class_name": "V8TopLevelFrameTaxiStarter"}]
IDemoBaseUIOSelectorClean = UIDesktop.UIOSelector_SearchUIONormalize_UIOSelector(IDemoBaseUIOSelectorDirty)
```

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_TryRestore_Dict(inSpecificationList) [\[исходный код\]](#)

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 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
# 1С: UIO Селектор выбора базы
IDemoBaseUIOSelector = [{"title": "Запуск 1С:Предприятия", "class_name": "V8TopLevelFrameTaxiStarter", "ba"}]
UIDesktop.UIOSelector_TryRestore_Dict(IDemoBaseUIOSelector) # Попытка восстановления свернутого окна n
```

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) [\[исходный код\]](#)

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 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
# 1С: UIO Селектор выбора базы
DemoBaseUIOSelector = [{"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)
- **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*) [\[исходный код\]](#)

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 СЕССИИ. 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
# 1С: 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)
- **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*) [\[исходный код\]](#)

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
# 1С: 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 помо
```

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.

Результат:

Список словарей - уровней UIO объектов

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

Установить фокус и выполнить подсветку UIO объекта на экране

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

Параметры:

- **IWrapperObject** (*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) \[исходный код\]
```

Получить индекс UIO объекта inElement в списке родительского UIO объекта.

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

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_Highlight(IWrapperObject, colour='green', thickness=2, fill=None, rect=None, inFlagSetFocus=False) \[исходный код\]
```

Выполнить подсветку UIO объекта на экране

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

Параметры:

- **IWrapperObject** (*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

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`. Прямоую 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 ознакомиться 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 разрядностью, 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](#)

.. v1.2.13 replace:: v1.2.13

⏪ Предыдущая

Следующая ⏩

© Copyright 2022, ООО "ОПЕН РПА".

Собрано при помощи [Sphinx](#) с использованием [темы](#), предоставленной [Read the Docs](#).



🏠 » 3. Функции UIWeb

3. Функции UIWeb

Общее

Документация к модулю будет опубликована в Июле 2022. По всем вопросам роботизации WEB приложений обращайтесь в центр поддержки клиентов pyOpenRPA.

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](#)

.. v1.2.13 replace:: v1.2.13

⏪ Предыдущая

Следующая ⏩

© Copyright 2022, ООО "ОПЕН РПА".

Собрано при помощи [Sphinx](#) с использованием [темы](#), предоставленной [Read the Docs](#).

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.

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

Functions:

<code>Down</code> (inKeyInt[, inWaitAfterSecFloat])	Нажать (опустить) клавишу.
<code>HotkeyCombination</code> (*inKeyList[, ...])	Получает перечень клавиш для одновременного нажатия.
<code>HotkeyCtrlA_ctrlC</code> ([inWaitAfterSecFloat])	Выполнить выделение текста, после чего скопировать его в буфер обмена ВНИМАНИЕ! НЕ ЗАВИСИТ ОТ ТЕКУЩЕЙ РАСКЛАДКИ КЛАВИАТУРЫ
<code>HotkeyCtrlV</code> ([inWaitAfterSecFloat])	Выполнить вставку текста из буфера обмена ВНИМАНИЕ! НЕ ЗАВИСИТ ОТ ТЕКУЩЕЙ РАСКЛАДКИ КЛАВИАТУРЫ
<code>IsDown</code> (inKeyInt)	Проверить, опущена ли клавиша.
<code>Send</code> (inKeyInt[, inDoPressBool, ...])	Имитация нажатия/отпускания любой физической клавиши.
<code>Up</code> (inKeyInt[, inWaitAfterSecFloat])	Отпустить (поднять) клавишу.
<code>Wait</code> (inKeyInt[, inWaitAfterSecFloat])	Блокирует осуществление программы, пока данная обозначенная клавиша не будет нажата.

`write (inTextStr[, inDelayFloat, ...])`

Печатает текст, который был передан в переменной `inTextStr` (поддерживает передачу в одной строке символов разного языка).

Data:

`KEY_HOT_COLON`

`KEY_RUS_Ж`

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

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!

```
# 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)` [\[исходный код\]](#)

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` [\[исходный код\]](#)

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 \[исходный код\]
```

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 \[исходный код\]
```

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_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 \[исходный код\]
```

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 отпущения 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 [\[исходный код\]](#)

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!

```
# 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*) [\[исходный код\]](#)

Блокирует осуществление программы, пока данная обозначенная клавиша не будет нажата.

ВНИМАНИЕ! НЕ ЗАВИСИТ ОТ ТЕКУЩЕЙ РАСКЛАДКИ КЛАВИАТУРЫ. ОЖИДАЕТ НАЖАТИЕ СООТВЕТСТВУЮЩЕЙ 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*) [\[исходный код\]](#)

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

```
# 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_Э = 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](#)

.. v1.2.13 replace:: v1.2.13

[← Предыдущая](#)

[Следующая →](#)

© Copyright 2022, ООО "ОПЕН РПА".

Собрано при помощи [Sphinx](#) с использованием [темы](#), предоставленной [Read the Docs](#).

5. Функции Clipboard

Описание функций

Functions:

<code>Get ()</code>	Получить текстовое содержимое буфера обмена.
<code>set (inTextStr)</code>	Установить текстовое содержимое в буфер обмена.

`pyOpenRPA.Robot.Clipboard.Get()` [\[исходный код\]](#)

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)` [\[исходный код\]](#)

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](#)

.. v1.2.13 replace:: v1.2.13

© Copyright 2022, ООО "ОПЕН РПА".

Собрано при помощи [Sphinx](#) с использованием темы, предоставленной [Read the Docs](#).

6. Функции Mouse

Общее

Описание функций

Functions:

<code>click</code> ((inXInt, inYInt, inClickCountInt, ...))	Нажатие (вниз) кнопки мыши и затем немедленно выпуск (вверх) её.
<code>clickDouble</code> ((inXInt, inYInt, ...))	Двойное нажатие левой клавиши мыши.
<code>Down</code> ((inXInt, inYInt, inButtonStr, ...))	Переместить указатель по координатам inXInt, inYInt, после чего нажать (вниз) клавишу мыши и не отпускать до выполнения соответствующей команды (см.
<code>MoveTo</code> ((inXInt, inYInt, ...))	Переместить указатель мыши на позицию inXInt, inYInt за время inMoveDurationSecFloat.
<code>ScrollHorizontal</code> (inScrollClickCountInt[, ...])	!ТОЛЬКО ДЛЯ LINUX! Переместить указатель мыши на позицию inXInt, inYInt и выполнить горизонтальную прокрутку (скроллинг) виртуальным колесом мыши на количество щелчков inScrollClickCountInt.
<code>ScrollVertical</code> (inScrollClickCountInt[, ...])	Переместить указатель мыши на позицию inXInt, inYInt и выполнить вертикальную прокрутку (скроллинг) колесом мыши на количество щелчков inScrollClickCountInt.
<code>up</code> ((inXInt, inYInt, inButtonStr, ...))	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]
```

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(б
```

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) \[исходный код\]
```

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(2
```

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) \[исходный код\]
```

Переместить указатель по координатам inXInt, inYInt, после чего нажать (вниз) клавишу мыши и не отпускать до выполнения соответствующей 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) \[исходный код\]
```

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) 200
```

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) [\[исходный код\]](#)

!ТОЛЬКО ДЛЯ LINUX! 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 координатам
Mouse.ScrollHorizontal(-100) #Крутить колесо мыши вверх на 100 кликов по текущим координатам указателя
```

Параметры:

- **inScrollClickCountInt** (*int, обязательный*) – Количество щелчков 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) [\[исходный код\]](#)

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 координатам
Mouse.ScrollVertical(-100) #Крутить колесо мыши вверх на 100 кликов по текущим координатам указателя
```

Параметры:

- **inScrollClickCountInt** (*int, обязательный*) – Количество щелчков 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) \[исходный код\]
```

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](#)

.. v1.2.13 replace:: v1.2.13

[← Предыдущая](#)

[Следующая →](#)

© Copyright 2022, ООО "ОПЕН РПА".

Собрано при помощи [Sphinx](#) с использованием [темы](#), предоставленной [Read the Docs](#).

8. 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.

Как запустить скрипт робота?

Запустить скрипт робота можно 2-мя способами:

- Скрипт Python (файл .py)
- Скрипт в Студии pyOpenRPA

Скрипт Python (файл .py)

Чтобы начать использовать модуль робота достаточно выполнить в файле скрипта соответствующие 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 роботизации 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 (Resources\WPpy32-3720python-3.7.2python.exe)
- Запустить из интерпретатора Python x64 (Resources\WPpy64-3720python-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 Python
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 интерпретатор Python
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 средства 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 директорию там, где находится этот .cmd файл
copy /Y ..\Resources\WPy32-3720\python-3.7.2\python.exe ..\Resources\WPy32-3720\OpenRPAOrche
.\..\Resources\WPy32-3720\python-3.7.2\OpenRPAOrchestrator.exe orchestratorMain.py # Выполнить инициализацию
pause >nul # Не закрывать окно консоли после завершения работы скрипта робота
```

Быстрая навигация

- [Сообщество ruOpenRPA \(telegram\)](#)
- [Сообщество ruOpenRPA \(tenchat\)](#)
- [Сообщество ruOpenRPA \(вконтакте\)](#)
- [Презентация ruOpenRPA](#)
- [Портал ruOpenRPA](#)
- [Репозиторий ruOpenRPA](#)

.. v1.2.13 replace:: v1.2.13

← Предыдущая

Следующая →

© Copyright 2022, ООО "ОПЕН РПА".

Собрано при помощи [Sphinx](#) с использованием [темы](#), предоставленной [Read the Docs](#).



[🏠](#) » 1. Описание

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](#)

pyOpenRPA - роботы помогут!

Быстрая навигация

- [Сообщество pyOpenRPA \(telegram\)](#)
- [Сообщество pyOpenRPA \(tenchat\)](#)
- [Сообщество pyOpenRPA \(вконтакте\)](#)
- [Презентация pyOpenRPA](#)
- [Портал pyOpenRPA](#)
- [Репозиторий pyOpenRPA](#)

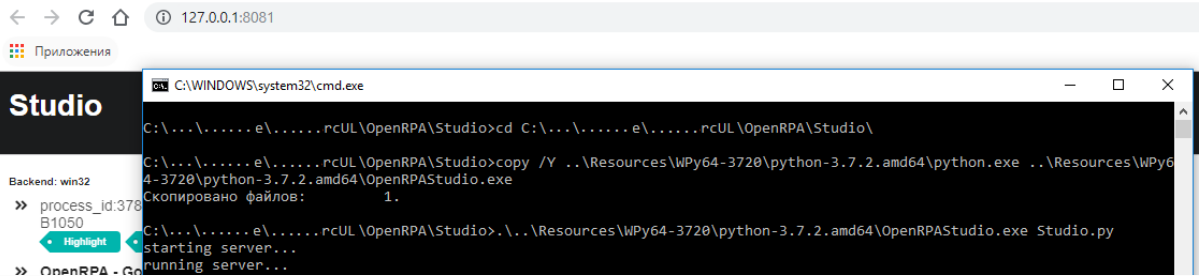
.. v1.2.13 replace:: v1.2.13

[← Предыдущая](#)

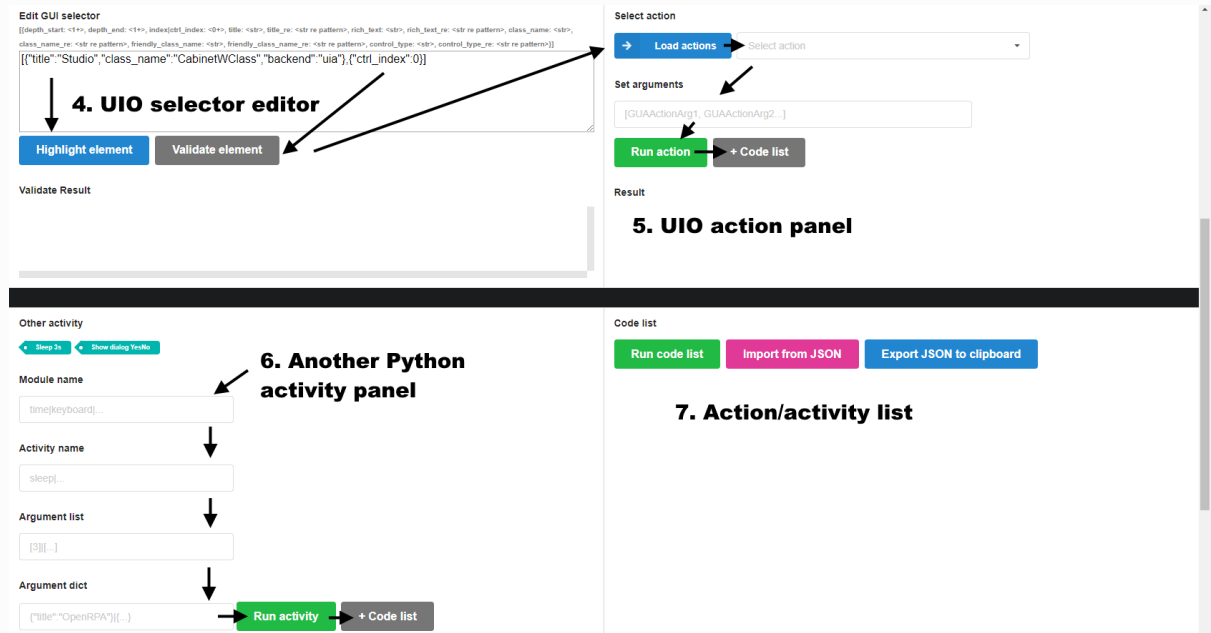
[Следующая →](#)

© Copyright 2022, ООО "ОПЕН РПА".

Собрано при помощи [Sphinx](#) с использованием [темы](#), предоставленной [Read the Docs](#).



Скриншот 2

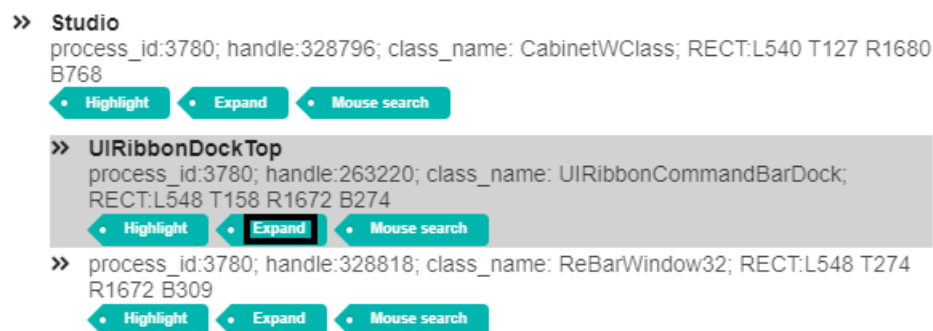


Извлечь UI дерево

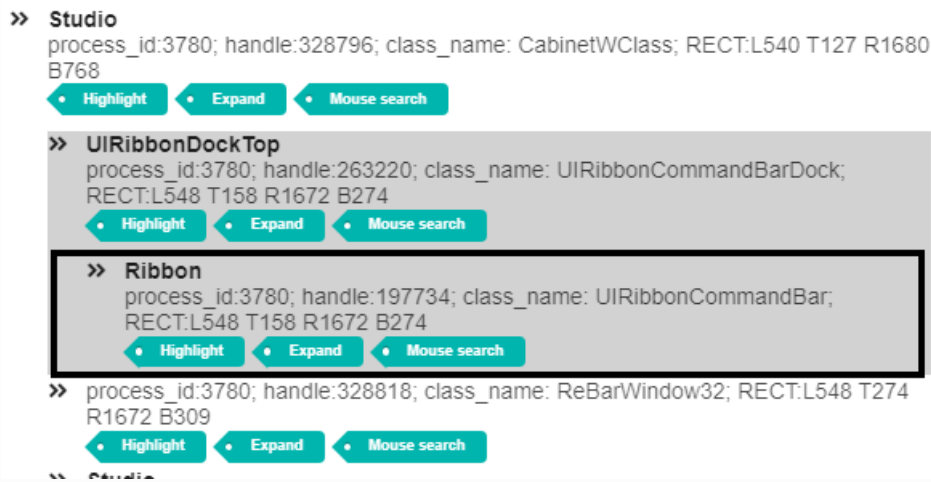
Чтобы извлечь дерево пользовательского интерфейса выполните следующие действия: в

[UI Tree viewer](#) выбрать интересующий UI объект и нажать кнопку [Expand](#).

Действие: Нажать по кнопке «Expand»



Итог

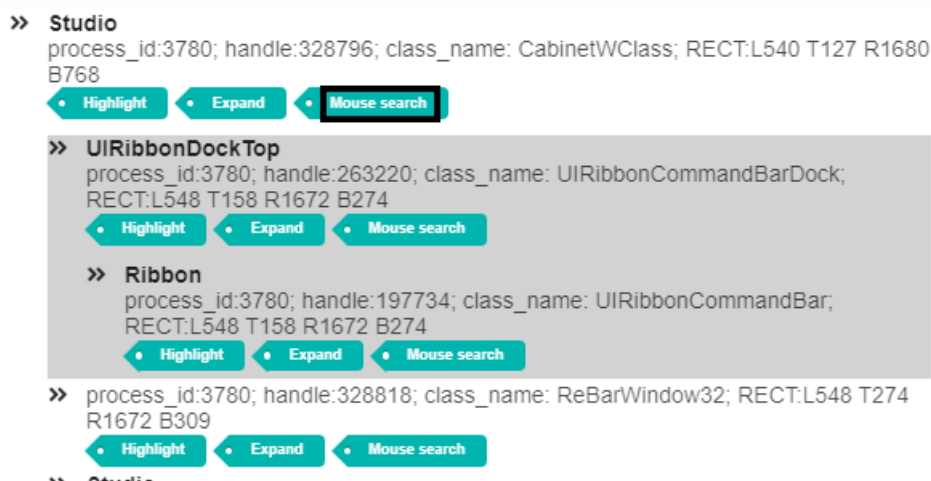


Поиск UI объекта по наведению мыши

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

Имя	Дата изменения	Тип	Размер
__pycache__	20.06.2019 20:13	Папка с файлами	
Reports	15.09.2019 9:49	Папка с файлами	
Web	14.07.2019 10:46	Папка с файлами	
JSONNormalize.py	09.06.2019 22:56	Файл "PY"	4 КБ
ProcessCommunicator.py	09.06.2019 22:56	Файл "PY"	8 КБ
PythonDebug_32	09.06.2019 22:56	Сценарий Windo...	1 КБ
Studio.py	09.06.2019 22:56	Файл "PY"	10 КБ
StudioRun_32	09.06.2019 22:56	Сценарий Windo...	1 КБ
StudioRun_64	09.06.2019 22:56	Сценарий Windo...	1 КБ
ValueVerify.py	09.06.2019 22:56	Файл "PY"	1 КБ

Итог: Интересующий UI объект будет отображен в [UI tree viewer](#)

Backend: uia

- >> **Studio**
process_id:3780; handle:328796; class_name: CabinetWClass; RECT:L540 T127 R1680 B768
 - Highlight
 - Expand
 - Mouse search
- >> **Studio**
process_id:3780; handle:2818786; class_name: ShellTabWindowClass; RECT:L548 T309 R1672 B760
 - Highlight
 - Expand
 - Mouse search
- >> process_id:3780; handle:132082; class_name: DUIViewWndClassName; RECT:L548 T309 R1672 B760
 - Highlight
 - Expand
 - Mouse search
- >> **Просмотр папок оболочки**
process_id:3780; handle:656482; class_name: DUIListView; RECT:L708 T309 R1672 B737
 - Highlight
 - Expand
 - Mouse search
- >> **Просмотр элементов**
process_id:3780; handle:787334; class_name: UIItemsView; RECT:L708 T309 R1672 B737
 - Highlight
 - Expand

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

Level 0:

```
{"title":"Studio","rich_text":"Studio","process_id":3780,"process":3780,"handle":328796,"class_name":"CabinetWClass","control_type":"Window","control_id":null,"rectangle":{"left":540,"top":127,"right":1680,"bottom":768},"backend":"uia"}
```

Click it!

Level 1:

```
{"title":"UIRibbonDockTop","rich_text":"UIRibbonDockTop","process_id":3780,"process":3780,"handle":263220,"class_name":"UIRibbonCommandBarDock","control_type":"Pane","control_id":null,"rectangle":{"left":548,"top":158,"right":1672,"bottom":274},"ctrl_index":0}
```

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 ruOpenRPA \(telegram\)](#)
- [Сообщество ruOpenRPA \(tenchat\)](#)
- [Сообщество ruOpenRPA \(вконтакте\)](#)
- [Презентация ruOpenRPA](#)
- [Портал ruOpenRPA](#)
- [Репозиторий ruOpenRPA](#)

.. v1.2.13 replace:: v1.2.13

[← Предыдущая](#)

[Следующая →](#)

© Copyright 2022, ООО "ОПЕН РПА".

Собрано при помощи [Sphinx](#) с использованием [темы](#), предоставленной [Read the Docs](#).

1. Описание

Общее

Модуль оркестратор - это координирующее звено, которое обеспечивает бесперебойную работу 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

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 подход 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 ruOpenRPA вы сможете существенно улучшить качество вашего бизнеса. Платформа роботизации ruOpenRPA - 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 ruOpenRPA. Контакты: [2. Лицензия & Контакты](#)

ruOpenRPA - роботы помогут!

Быстрая навигация

- [Сообщество ruOpenRPA \(telegram\)](#)
- [Сообщество ruOpenRPA \(tenchat\)](#)
- [Сообщество ruOpenRPA \(вконтакте\)](#)
- [Презентация ruOpenRPA](#)
- [Портал ruOpenRPA](#)
- [Репозиторий ruOpenRPA](#)

.. v1.2.13 replace:: v1.2.13

[← Предыдущая](#)

[Следующая →](#)

© Copyright 2022, ООО "ОПЕН РПА".

Собрано при помощи [Sphinx](#) с использованием [темы](#), предоставленной [Read the Docs](#).

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 развернут 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.

Functions:

<code>ActivityItemCreate</code> (inDef[, inArgList, ...])	Создать Активность (ActivityItem).
<code>ActivityItemDefAliasCreate</code> (inDef[, ...])	Создать синоним (текстовый ключ) для инициации выполнения функции в том случае, если запрос на выполнения пришел из вне (передача функций невозможна).
<code>ActivityItemDefAliasModulesLoad</code> ()	Загрузить все функции из импортированных модулей sys.modules в ActivityItem синонимы - полезно для отладки со стороны панели управления.
<code>ActivityItemDefAliasUpdate</code> (inDef, inAliasStr)	Обновить синоним (текстовый ключ) для инициации выполнения функции в том случае, если запрос на выполнения пришел из вне (передача функций невозможна).
<code>ActivityItemHelperDefAutofill</code> (inDef)	Анализ аргументов функции по синониму (текстовому ключу).
<code>ActivityItemHelperDefList</code> ([inDefQueryStr])	Получить список синонимов (текстовых ключей), доступных для использования в Активностях (ActivityItem).
<code>AgentActivityItemAdd</code> (inHostNameStr, ...[, ...])	Добавить активность в словарь активностей выбранного Агента
<code>AgentActivityItemExists</code> (inHostNameStr, ...)	Выполнить проверку, что активность (ActivityItem) была отправлена на сторону Агента.

<code>AgentActivityItemReturnExists</code> (inGUIDStr[, ...])	Выполнить проверку, что активность (ActivityItem) была выполнена на стороне Агента и результат был получен на стороне Оркстратора.
<code>AgentActivityItemReturnGet</code> (inGUIDStr[, ...])	Ожидает появления результата по активности (ActivityItem).
<code>AgentOSCMD</code> (inHostNameStr, inUserStr, inCMDStr)	Отправка команды командной строки на сессию, где работает ruOpenRPA.Agent.
<code>AgentOSFileBinaryDataBase64StrAppend</code> (...[, ...])	Добавить бинарную информацию в существующий бинарный файл, который будет расположен по адресу inFilePathStr на стороне Агента с содержимым, декодированным с формата base64: inFileDataBase64Str
<code>AgentOSFileBinaryDataBase64StrCreate</code> (...[, ...])	Создать бинарный файл, который будет расположен по адресу inFilePathStr на стороне Агента с содержимым, декодированным с формата base64: inFileDataBase64Str
<code>AgentOSFileBinaryDataBase64StrReceive</code> (...[, ...])	Выполнить чтение бинарного файла и получить содержимое в формате base64 (строка)
<code>AgentOSFileBinaryDataBytesCreate</code> (...[, ...])	Создать бинарный файл, который будет расположен по адресу inFilePathStr на стороне Агента с содержимым inFileDataBytes
<code>AgentOSFileBinaryDataReceive</code> (inHostNameStr, ...)	Чтение бинарного файла на стороне Агента по адресу inFilePathStr.
<code>AgentOSFileSend</code> (inHostNameStr, inUserStr, ...)	Отправить файл по адресу inOrchestratorFilePathStr со стороны Оркстратора и сохранить по адресу inAgentFilePathStr на стороне Агента.
<code>AgentOSFileTextDataStrCreate</code> (inHostNameStr, ...)	Создать текстовый файл, который будет расположен по адресу inFilePathStr на стороне Агента с содержимым inFileDataStr в кодировке inEncodingStr
<code>AgentOSFileTextDataStrReceive</code> (inHostNameStr, ...)	Чтение текстового файла на стороне Агента по адресу inFilePathStr.
<code>AgentOSLogoff</code> (inHostNameStr, inUserStr)	Выполнить операцию logoff на стороне пользователя.
<code>AgentProcessWOExeUpperUserListGet</code> (...[, ...])	Получить список процессов, которые выполняются на сессии Агента.
<code>GSettingsGet</code> ([inGSettings])	Вернуть глобальный словарь настроек Оркстратора.
<code>GSettingsKeyListValueAppend</code> (inValue[, ...])	Применить операцию .append к объекту, расположенному по адресу списка ключей inKeyList в глобальном словаре настроек Оркстратора GSettings.
<code>GSettingsKeyListValueGet</code> ([inKeyList, ...])	Получить значение из глобального словаря настроек Оркстратора GSettings по списку ключей.
<code>GSettingsKeyListValueOperatorPlus</code> (inValue[, ...])	Применить оператор сложения (+) к объекту, расположенному по адресу списка ключей inKeyList в глобальном словаре настроек Оркстратора GSettings.
<code>GSettingsKeyListValueSet</code> (inValue[, ...])	Установить значение из глобального словаря настроек Оркстратора GSettings по списку ключей.
<code>osCMD</code> (inCMDStr[, inRunAsyncBool, inLogger])	Отправить команду на выполнение на сессию, где выполняется Оркстратор.
<code>OSCredentialsVerify</code> (inUserStr, inPasswordStr)	Выполнить верификацию доменного (локального) пользователя по паре логин/пароль
<code>OSLogoff</code> ()	Выполнить отключение сессии, на которой выполняется Оркстратор.
<code>OSRemotePCRestart</code> (inHostStr[, inForceBool, ...])	Отправить сигнал на удаленную перезагрузку операционной системы.
<code>orchestrator</code> ([inGSettings, ...])	Инициализация ядра Оркстратора (всех потоков)
<code>OrchestratorInitWait</code> ()	Ожидать инициализацию ядра Оркстратора

<code>OrchestratorIsAdmin ()</code>	Проверить, запущен ли Оркестратор с правами администратора.
<code>OrchestratorIsInited ()</code>	Проверить, было ли проинициализировано ядро Оркестратора
<code>OrchestratorLoggerGet ()</code>	Получить логгер Оркестратора
<code>OrchestratorPySearchInit (inGlobPatternStr[, ...])</code>	Выполнить поиск и инициализацию пользовательских .py файлов в Оркестраторе (например панелей управления роботом)
<code>OrchestratorRerunAsAdmin ()</code>	Перезапустить Оркестратор с правами локального администратора.
<code>OrchestratorRestart ([inGSettings])</code>	Перезапуск Оркестратора с сохранением информации о запущенных RDP сессиях.
<code>OrchestratorScheduleGet ()</code>	Базовый объект расписания, который можно использовать для запуска / остановки роботов.
<code>OrchestratorSessionRestore ([inGSettings])</code>	Восстановить состояние Оркестратора, если ранее состояние Оркестратора было сохранено с помощью функции <code>OrchestratorSessionSave</code> :
<code>OrchestratorSessionSave ([inGSettings])</code>	Сохранить состояние Оркестратора (для дальнейшего восстановления в случае перезапуска):
<code>OrchestratorThreadStart (inDef, *inArgList, ...)</code>	Запустить функцию в отдельном потоке.
<code>ProcessDefIntervalCall (inDef, inIntervalSecFloat)</code>	Периодический вызов функции Python.
<code>ProcessIsStarted (inProcessNameWOExeStr)</code>	Проверить, запущен ли процесс, который в наименовании содержит <code>inProcessNameWOExeStr</code> .
<code>ProcessListGet ([inProcessNameWOExeList])</code>	Вернуть список процессов, запущенных на ОС, где работает Оркестратор.
<code>ProcessStart (inPathStr, inArgList[, ...])</code>	Запуск процесса на сессии Оркестратора, если на ОС не запущен процесс <code>inStopProcessNameWOExeStr</code> .
<code>ProcessStop (inProcessNameWOExeStr, ...[, ...])</code>	Остановить процесс на ОС, где работает Оркестратор, под учетной записью <code>inUserNameStr</code> .
<code>ProcessorActivityItemAppend ([inGSettings, ...])</code>	Добавить активность (<code>ActivityItem</code>) в процессорную очередь.
<code>ProcessorActivityItemCreate (inDef[, ...])</code>	Создать Активность (<code>ActivityItem</code>).
<code>ProcessorAliasDefCreate (inDef[, inAliasStr, ...])</code>	Создать синоним (текстовый ключ) для инициации выполнения функции в том случае, если запрос на выполнения пришел из вне (передача функций невозможна).
<code>ProcessorAliasDefUpdate (inDef, inAliasStr[, ...])</code>	Обновить синоним (текстовый ключ) для инициации выполнения функции в том случае, если запрос на выполнения пришел из вне (передача функций невозможна).
<code>PythonStart (inModulePathStr, inDefNameStr[, ...])</code>	Импорт модуля и выполнение функции в процессе Оркестратора.
<code>RDPSessionCMDRun (inRDPSessionKeyStr, inCMDStr)</code>	Отправить CMD команду на удаленную сессию через RDP окно (без Агента).
<code>RDPSessionConnect (inRDPSessionKeyStr[, ...])</code>	Выполнить подключение к RDP и следить за стабильностью соединения со стороны Оркестратора.
<code>RDPSessionDisconnect (inRDPSessionKeyStr[, ...])</code>	Выполнить отключение RDP сессии и прекратить мониторить его активность.
<code>RDPSessionFileStoredRecieve (...[, inGSettings])</code>	Получение файла со стороны RDP сессии на сторону Оркестратора через UI инструменты RDP окна (без Агента).
<code>RDPSessionFileStoredSend (inRDPSessionKeyStr, ...)</code>	Отправка файла со стороны Оркестратора на сторону RDP сессии через UI инструменты RDP окна (без Агента).

<code>RDPSessionLogoff</code> (inRDPSessionKeyStr[, ...])	Выполнить отключение (logoff) на RDP сессии и прекратить мониторить активность со стороны Оркестратора.
<code>RDPSessionMonitorStop</code> (inRDPSessionKeyStr[, ...])	Прекратить мониторить активность RDP соединения со стороны Оркестратора.
<code>RDPSessionProcessStartIfNotRunning</code> ([...], [...])	Выполнить запуск процесса на RDP сессии через графические UI инструменты (без Агента).
<code>RDPSessionProcessStop</code> (inRDPSessionKeyStr, ...)	Отправка CMD команды в RDP окне на остановку процесса (без Агента).
<code>RDPSessionReconnect</code> (inRDPSessionKeyStr[, ...])	Выполнить переподключение RDP сессии и продолжить мониторить его активность.
<code>RDPTemplateCreate</code> (inLoginStr, inPasswordStr)	Создать шаблон подключения RDP (dict).
<code>SchedulerActivityTimeAddWeekly</code> ([...])	Добавить активность по расписанию.
<code>StorageRobotExists</code> (inRobotNameStr)	Проверить, существует ли ключ inRobotNameStr в хранилище пользовательской информации StorageDict (GSettings > StarageDict)
<code>StorageRobotGet</code> (inRobotNameStr)	Получить содержимое по ключу робота inRobotNameStr в хранилище пользовательской информации StorageDict (GSettings > StarageDict)
<code>UACKeyListCheck</code> (inRequest, inRoleKeyList)	Проверить права доступа для пользователя запроса по списку ключей до права.
<code>UACSuperTokenUpdate</code> (inSuperTokenStr[, ...])	Добавить супертокен (полный доступ).
<code>UACUpdate</code> (inADLoginStr[, inADStr, ...])	Дообогачение словаря доступа UAC пользователя inADStrinADLoginStr.
<code>UACUserDictGet</code> (inRequest)	Вернуть UAC (User Access Control) словарь доступов для пользователя, который отправил запрос.
<code>WebAuditMessageCreate</code> ([inRequest, ...])	Создание сообщения ИТ аудита с такими сведениями как (Домен, IP, логин и тд.).
<code>WebCPUUpdate</code> (inCPKeyStr[, inHTMLRenderDef, ...])	Добавить панель управления робота в Оркестратор.
<code>WebListenCreate</code> ([inServerKeyStr, ...])	Настроить веб-сервер Оркестратора.
<code>WebRequestGet</code> ()	Вернуть экземпляр HTTP запроса, если функция вызвана в потоке, который был порожден для отработки HTTP запроса пользователя.
<code>WebRequestParseBodyBytes</code> ([inRequest])	Извлечь данные в байт виде из тела (body) HTTP запроса.
<code>WebRequestParseBodyJSON</code> ([inRequest])	Извлечь из тела (body) запроса HTTP JSON данные и преобразовать в Dict / List структуры языка Python.
<code>WebRequestParseBodyStr</code> ([inRequest])	Извлечь данные в виде строки из тела (body) HTTP запроса.
<code>WebRequestParseFile</code> ([inRequest])	Извлечь файл (наименование + содержимое в виде строки байт b ^{""}) из HTTP запроса пользователя.
<code>WebRequestParsePath</code> ([inRequest])	Извлечь декодированный URL путь из HTTP запроса пользователя в формате строки.
<code>WebRequestResponseSend</code> (inResponseStr[, ...])	Установить ответ на HTTP запрос пользователя.
<code>WebURLConnectDef</code> (inMethodStr, inURLStr, ...)	Подключить функцию Python к URL.
<code>WebURLConnectFile</code> (inMethodStr, inURLStr, ...)	Подключить файл к URL.
<code>WebURLConnectFolder</code> (inMethodStr, inURLStr, ...)	Подключить папку к URL.
<code>WebURLIndexChange</code> ([inURLIndexStr])	Изменить адрес главной страницы Оркестратора.
<code>WebUserInfoGet</code> ([inRequest])	Информация о пользователе, который отправил HTTP запрос.

<code>WebUserIsSuperToken</code> ([inRequest, inGSettings])	Проверить, авторизован ли HTTP запрос с помощью супер токена (токен, который не истекает).
<code>WebUserUACHierarchyGet</code> ([inRequest])	Вернуть словарь доступа UAC в отношении пользователя, который выполнил HTTP запрос inRequest

pyOpenRPA.Orchestrator.__Orchestrator__.ActivityItemCreate(inDef, inArgList=None, inArgDict=None, inArgGSettingsStr=None, inArgLoggerStr=None, inGUIDStr=None, inThreadBool=False) [\[исходный код\]](#)

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) [\[исходный код\]](#)

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() [\[исходный код\]](#)

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) [\[исходный код\]](#)

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)` [\[исходный код\]](#)

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)`
[\[исходный код\]](#)

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)` [\[исходный код\]](#)

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)` [\[исходный код\]](#)

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]

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) [исходный код]

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

Результат:

Результат выполнения активности. !ВНИМАНИЕ! Возвращаются только те 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]

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; False - Не отправлять; Default True

- **inCMDEncodingStr** – Кодировка DOS среды, в которой выполняется 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:

GUID (GUID) строка Активности (ActivityItem). Далее можно ожидать результат этой функции по GUID с помощью функции AgentActivityItemReturnGet

```
pyOpenRPA.Orchestrator.__Orchestrator__.AgentOSFileBinaryDataBase64StrAppend(inHostNameStr, inUserStr, inFilePathStr, inFileDataBase64Str, inGSettings=None) \[исходный код\]
```

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:

GUID (GUID) строка Активности (ActivityItem). Далее можно ожидать результат этой функции по GUID с помощью функции AgentActivityItemReturnGet

```
pyOpenRPA.Orchestrator.__Orchestrator__.AgentOSFileBinaryDataBase64StrCreate(inHostNameStr, inUserStr, inFilePathStr, inFileDataBase64Str, inGSettings=None) \[исходный код\]
```

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:

GUID (GUID) строка Активности (ActivityItem). Далее можно ожидать результат этой функции по GUID с помощью функции AgentActivityItemReturnGet

```
pyOpenRPA.Orchestrator.__Orchestrator__.AgentOSFileBinaryDataBase64StrReceive(inHostNameStr, inUserStr, inFilePathStr, inGSettings=None) \[исходный код\]
```

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:

GUID (GUID) строка Активности (ActivityItem). Далее можно ожидать результат этой функции по GUID с помощью функции AgentActivityItemReturnGet

```
pyOpenRPA.Orchestrator.__Orchestrator__.AgentOSFileBinaryDataBytesCreate(inHostNameStr, inUserStr, inFilePathStr, inFileDataBytes, inGSettings=None) [исходный код]
```

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:

GUID (GUID) строка Активности (ActivityItem). Далее можно ожидать результат этой функции по GUID с помощью функции AgentActivityItemReturnGet

```
pyOpenRPA.Orchestrator.__Orchestrator__.AgentOSFileBinaryDataReceive(inHostNameStr, inUserStr, inFilePathStr) [исходный код]
```

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]
```

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 исполнение. Если вам нужен результат функции, то необходимо сначала убедиться в том, что ядро Оркестратора было инициализировано (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) \[исходный код\]
```

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) \[исходный код\]
```

Чтение текстового файла на стороне Агента по адресу `inFilePathStr`. По ГУИД 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)` [\[исходный код\]](#)

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)` [\[исходный код\]](#)

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)` [\[исходный код\]](#)

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)` [\[исходный код\]](#)

Применить операцию .append к объекту, 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*)
[\[исходный код\]](#)

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*)
[\[исходный код\]](#)

Применить оператор сложения (+) к объекту, 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*)
[\[исходный код\]](#)

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*)

[\[исходный код\]](#)

Отправить команду на выполнение на сессию, где выполняется Оркестратор.

Поддерживается: Windows: +, Linux: +

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=""*)

[\[исходный код\]](#)

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() [\[исходный код\]](#)

Выполнить отключение сессии, на которой выполняется Оркестратор.

Результат:

pyOpenRPA.Orchestrator.__Orchestrator__.OSRemotePCRestart(*inHostStr, inForceBool=True, inLogger=None*)

[\[исходный код\]](#)

Отправить сигнал на удаленную перезагрузку операционной системы.

!ВНИМАНИЕ! Перезапуск будет принят, если учетная запись имеет полномочия на перезапуск на соответствующей 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__.Orchestrator(inGSettings=None, inDumpRestoreBool=True, inRunAsAdministratorBool=True) \[исходный код\]
```

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 администратора и обеспечить их; False - Не обеспечивать права администратора

```
pyOpenRPA.Orchestrator.__Orchestrator__.OrchestratorInitWait() → None \[исходный код\]
```

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() \[исходный код\]
```

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__.OrchestratorIsInited() → bool \[исходный код\]
```

Проверить, было ли проинициализировано ядро Оркестратора

Результат:

True - Ядро Оркестратора было проинициализировано; False - Требуется время на инициализацию

Тип результата:

bool

```
pyOpenRPA.Orchestrator.__Orchestrator__.OrchestratorLoggerGet() → logging.Logger \[исходный код\]
```

Получить логгер Оркестратора

Результат:

Логгер

```
pyOpenRPA.Orchestrator.__Orchestrator__.OrchestratorPySearchInit(inGlobPatternStr, inDefStr=None, inDefArgNameGSettingsStr=None, inAsyncInitBool=False) \[исходный код\]
```

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 ru без вызова каких-либо функций внутри)
# автоинициализация всех .ру файлов, с префиксом CP_, которые расположены в папке ControlPanel
Orchestrator.OrchestratorPySearchInit(inGlobPatternStr="ControlPanel\CP_*.py")

# ВАРИАНТ ИСПОЛЬЗОВАНИЯ 2 (инициализация модуля ru с вызовом функции внутри) - преимущественно для обр
# автоинициализация всех .ру 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 .ру 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 ru модулей в отдельных параллельных потоках - псевдопараллельное выполнение. False - последовательная инициализация

Результат:

Сведения об инициализированных модулях в структуре: { «ModuleNameStr»: { «PyPathStr»: «», «Module»: ...}, ...}

pyOpenRPA.Orchestrator.__Orchestrator__.OrchestratorRerunAsAdmin() [\[исходный код\]](#)

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) [\[исходный код\]](#)

Перезапуск Оркестратора с сохранением информации о запущенных RDP сессиях.

Параметры:

inGSettings – Глобальный словарь настроек Оркестратора (синглтон)

pyOpenRPA.Orchestrator.__Orchestrator__.OrchestratorScheduleGet() → <module „schedule“ from ;c:\abs\archive\scopesrcul\pyopenrpa\resources\wpy64-3720\python-3.7.2.amd64\lib\site-packages\schedule_init_.py"> [\[исходный код\]](#)

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(1Process.StatusCheckStart)

#Многопоточный schedule. см. onucanie Orchestrator.OrchestratorThreadStart
Orchestrator.OrchestratorScheduleGet().every(5).seconds.do(Orchestrator.OrchestratorThreadStart,1Proce

```

Результат:

schedule объект

pyOpenRPA.Orchestrator.__Orchestrator__.OrchestratorSessionRestore(*inGSettings=None*)

[\[исходный код\]](#)

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*)

[\[исходный код\]](#)

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*)

[\[исходный код\]](#)

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 экземпляра

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

[\[исходный код\]](#)

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) [\[исходный код\]](#)

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) [\[исходный код\]](#)

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) [\[исходный код\]](#)

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%') [\[исходный код\]](#)

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 сессию - 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]
```

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** – GUID идентификатор активности (ActivityItem). Если GUID 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 GUID активности (ActivityItem)

```
pyOpenRPA.Orchestrator.__Orchestrator__.ProcessorActivityItemCreate(inDef, inArgList=None, inArgDict=None, inArgGSettingsStr=None, inArgLoggerStr=None, inGUIDStr=None, inThreadBool=False) [исходный код]
```

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) [\[исходный код\]](#)

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) [\[исходный код\]](#)

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)` [\[исходный код\]](#)

Импорт модуля и выполнение функции в процессе Оркестратора.

! Примечание

Импорт модуля `inModulePathStr` будет происходить каждый раз в вызове 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)` [\[исходный код\]](#)

Отправить CMD команду на удаленную сессию 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 соответствии 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]
```

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]
```

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*,
inRDPPFilePathStr, *inHostFilePathStr*, *inGSettings=None*) [\[исходный код\]](#)

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",
    inRDPPFilePathStr = "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
- **inRDPPFilePathStr** – 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, *inRDPPFilePathStr*, *inGSettings=None*) [\[исходный код\]](#)

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) \[исходный код\]
```

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)
\[исходный код\]
```

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)` [\[исходный код\]](#)

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)` [\[исходный код\]](#)

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)` [\[исходный код\]](#)

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]
```

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»: «77777sdfsdf77777sdfsdf77777777»,
  «SessionIsWindowExistBool»: False, «SessionIsWindowResponsibleBool»: False,
  «SessionIsIgnoredBool»: False
}
```

pyOpenRPA.Orchestrator.__Orchestrator__.SchedulerActivityTimeAddWeekly(inTimeHHMMStr='23:55', inWeekdayList=None, inActivityList=None, inGSettings=None) [\[исходный код\]](#)

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) [\[исходный код\]](#)

Проверить, существует ли ключ inRobotNameStr в хранилище пользовательской информации StorageDict (GSettings > StorageDict)

Параметры:

inRobotNameStr – Наименование (ключ) робота. !ВНИМАНИЕ! Наименование чувствительно к регистру

Результат:

True - ключ робота присутствует в хранилище; False - отсутствует

pyOpenRPA.Orchestrator.__Orchestrator__.StorageRobotGet(inRobotNameStr) [\[исходный код\]](#)

Получить содержимое по ключу робота inRobotNameStr в хранилище пользовательской информации StorageDict (GSettings > StorageDict)

Параметры:

inRobotNameStr – Наименование (ключ) робота. !ВНИМАНИЕ! Наименование чувствительно к регистру

Результат:

Значение или структура, которая расположена по адресу (GSettings > StorageDict > inRobotNameStr)

pyOpenRPA.Orchestrator.__Orchestrator__.UACKeyListCheck(inRequest, inRoleKeyList) → bool

[\[исходный код\]](#)

Проверить права доступа для пользователя запроса по списку ключей до права.

```
# ВАРИАНТ ИСПОЛЬЗОВАНИЯ 1 (инициализация модуля ru 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)

[\[исходный код\]](#)

Добавить супертокен (полный доступ). Супертокены используются для подключения к Оркстратору по API

Параметры:

- **inGSettings** – Глобальный словарь настроек Оркстратора (синглтон)
- **inSuperTokenStr** – Кодовая строковая комбинация, которая будет предоставлять доступ роботу / агенту для взаимодействия с Оркстратором по API

pyOpenRPA.Orchestrator.__Orchestrator__.UACUpdate(inADLoginStr, inADStr="", inADIsDefaultBool=True, inURLList=None, inRoleHierarchyAllowedDict=None, inGSettings=None) [\[исходный код\]](#)

Дообогачение словаря доступа UAC пользователя inADStrinADLoginStr. Если ранее словарь не устанавливался, то по умолчанию он {}. Далее идет дообогачение теми ключами, которые присутствуют в inRoleHierarchyAllowedDict

Параметры:

- **inGSettings** – Глобальный словарь настроек Оркстратора (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` [\[исходный код\]](#)

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__.WebAuditMessageCreate(inRequest=None, inOperationCodeStr='-', inMessageStr='-')` [\[исходный код\]](#)

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:

- **inRequest** – Экземпляр HTTP request. Опционален, если сообщение фиксируется из под потока, который был инициирован запросом пользователя
- **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__.WebCPUpdate(inCPKeyStr, inHTMLRenderDef=None, inJSONGeneratorDef=None, inJSInitGeneratorDef=None, inGSettings=None)` [\[исходный код\]](#)

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="", inPortInt=80, inCertFilePEMPathStr=None, inKeyFilePathStr=None, inGSettings=None) \[исходный код\]
```

Настроить веб-сервер Оркестратора.

Параметры:

- **inAddressStr** – IP адрес для прослушивания. Если «», 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 80. 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() \[исходный код\]
```

Вернуть экземпляр HTTP запроса, если функция вызвана в потоке, который был порожден для отработки HTTP запроса пользователя.

```
pyOpenRPA.Orchestrator.__Orchestrator__.WebRequestParseBodyBytes(inRequest=None) \[исходный код\]
```

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) \[исходный код\]
```

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)` [\[исходный код\]](#)

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)` [\[исходный код\]](#)

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)` [\[исходный код\]](#)

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(inResponseStr, inRequest=None, inContentTypeStr: Optional[str] = None, inHeadersDict: Optional[dict] = None)` [\[исходный код\]](#)

Установить ответ на HTTP запрос пользователя.

Параметры:

- **inResponseStr** – 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)` [\[исходный код\]](#)

Подключить функцию Python к URL.

Параметры:

- **inMethodStr** – Метод доступа по URL «GET» || «POST»
- **inURLStr** – URL адрес. Пример «/index»
- **inMatchTypeStr** – Тип соответствия 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** – MIME 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) \[исходный код\]
```

Подключить файл к URL.

Параметры:

- **inMethodStr** – Метод доступа по URL «GET» || «POST»
- **inURLStr** – URL адрес. Пример «/index»
- **inMatchTypeStr** – Тип соответствия 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, inGSettings=None, inUACBool=None, inUseCacheBool=False) \[исходный код\]
```

Подключить папку к URL.

Параметры:

- **inMethodStr** – Метод доступа по URL «GET» || «POST»
- **inURLStr** – URL адрес. Пример «/index»
- **inMatchTypeStr** – Тип соответствия строки URL с inURLStr: «BeginWith» || «Contains» || «Equal» || «EqualCase» || «EqualNoParam»
- **inFolderPathStr** – Путь к папке на диске, в которой искать файл и возвращать пользователю по HTTP
- **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 = '/') \[исходный код\]
```

Изменить адрес главной страницы Оркестратора. По умолчанию „/“

Параметры:

inURLIndexStr (*str, опционально*) – Новый адрес главной страницы Оркестратора.

```
pyOpenRPA.Orchestrator.__Orchestrator__.WebUserInfoGet(inRequest=None) \[исходный код\]
```

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»:

`pyOpenRPA.Orchestrator.__Orchestrator__.WebUserIsSuperToken(inRequest=None, inGSettings=None)`

[\[исходный код\]](#)

Проверить, авторизован ли HTTP запрос с помощью супер токена (токен, который не истекает).

Параметры:

- `inRequest` – Экземпляр HTTP request. Опционален, если сообщение фиксируется из под потока, который был инициирован запросом пользователя
- `inGSettings` – Глобальный словарь настроек Оркестратора (синглтон)

Результат:

True - является супертокеном; False - не является супертокеном

`pyOpenRPA.Orchestrator.__Orchestrator__.WebUserUACHierarchyGet(inRequest=None)` [\[исходный код\]](#)

Вернуть словарь доступа UAC в отношении пользователя, который выполнил HTTP запрос inRequest

Параметры:

`inRequest` – Экземпляр HTTP request. Опционален, если сообщение фиксируется из под потока, который был инициирован запросом пользователя

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](#)

.. v1.2.13 replace:: v1.2.13

[← Предыдущая](#)

[Следующая →](#)


```

# "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 algorithms
"ActivityList": []
},
"SchedulerDict": {
"Schedule": schedule, # https://schedule.readthedocs.io/en/stable/examples.html
"CheckIntervalSecFloat": 5.0, # Check interval in seconds
"ActivityTimeList": [
# {
# "Time": "00:00", # Time Full URL to test...

```

```

# "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(), mAgentUserNameS
"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 session 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 (reciev
        #     "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("Reports"):
            os.makedirs("Reports")
        #####
        # 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(
                "Reports\\" + datetime.datetime.now().strftime("%Y_%m_%d") + ".log")
            mRobotLoggerFormatter = logging.Formatter('%(asctime)s - %(levelname)s - %(message)s')
            mRobotLoggerFH.setFormatter(mRobotLoggerFormatter)
            # add handler to logger object
            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["Client"],
            #                                inKeyStr="DumpLogList",
            #                                inHashKeyStr="DumpLogListHashStr",
            #                                inRowCountInt=lResult["DumpLogListCountInt"],
            #                                mRobotLoggerFormatter)
            #mHandlerDumpLogList.setFormatter(mRobotLoggerFormatter)
            #mRobotLogger.addHandler(mHandlerDumpLogList)
        else:
            if lL: lL.warning("Pay attention! Your code has been call SettingsTemplate.Create - since pyO
    return lResult # return the result dict

```

Быстрая навигация

- [Сообщество pyOpenRPA \(telegram\)](#)
- [Сообщество pyOpenRPA \(tenchat\)](#)
- [Сообщество pyOpenRPA \(вконтакте\)](#)
- [Презентация pyOpenRPA](#)
- [Портал pyOpenRPA](#)
- [Репозиторий pyOpenRPA](#)

.. v1.2.13 replace:: v1.2.13

© Copyright 2022, ООО "ОПЕН РПА".

Собрано при помощи [Sphinx](#) с использованием темы, предоставленной [Read the Docs](#).

4. Как использовать?

Как запустить?

Хотите выполнить запуск Оркестратора?

Для этого достаточно (выбрать одно):

- запустить .cmd файл, расположенный в папке pyOpenRPA по адресу:
OrchestratorpyOpenRPA.Orchestrator_x64.cmd
- в свой .ру 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)
```

Параметры настройки

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 functionality
from pyOpenRPA.Tools import CrossOS
from pyOpenRPA import Orchestrator # Import orchestrator main
#Run as administrator
if not Orchestrator.OrchestratorIsAdmin():
    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__()
    Orchestrator.UACUpdate(inGSettings=gSettings, inADLoginStr="ND", inADStr="", inADIsDefaultBool=True,
    # TEST Add User IMasLov - Add Login IMasLov to superuser of the Orchestrator
    Orchestrator.UACUpdate(inGSettings=gSettings, inADLoginStr="IMaslov", 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)
    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="ControlPanel\\CP_*.py", inDefStr
    # Call the orchestrator def
    Orchestrator.Orchestrator(inGSettings=gSettings, inDumpRestoreBool=False)

```

Быстрая навигация

- [Сообщество ruOpenRPA \(telegram\)](#)
- [Сообщество ruOpenRPA \(tenchat\)](#)
- [Сообщество ruOpenRPA \(вконтакте\)](#)
- [Презентация ruOpenRPA](#)
- [Портал ruOpenRPA](#)
- [Репозиторий ruOpenRPA](#)

.. v1.2.13 replace:: v1.2.13

← Предыдущая

Следующая →

© Copyright 2022, ООО "ОПЕН РПА".

Собрано при помощи [Sphinx](#) с использованием [темы](#), предоставленной [Read the Docs](#).

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](#)

.. v1.2.13 replace:: v1.2.13

⏪ Предыдущая

Следующая ⏩

© Copyright 2022, ООО "ОПЕН РПА".

Собрано при помощи [Sphinx](#) с использованием темы, предоставленной [Read the Docs](#).