RobotFramework_Testsuites

v. 0.2.3

Mai Dinh Nam Son

30.11.2022

Contents

1 Introduction **1**
1.1 RobotFramework AIO testsuites management documentation . . . 1

2 Description **2**
2.1 Getting Started . . . 2
2.1.1 How to install . . . 2
2.2 Features . . . 3
2.2.1 Using configuration files in Json format . . . 3
2.2.2 Define 4 levels of configuration . . . 3
2.2.3 **Local configuration** . . . 4
2.2.4 **Access to configuration parameters** . . . 5

3 CConfig.py **6**
3.1 Class: dotdict . . . 6
3.2 Class: CConfig . . . 6
3.2.1 Method: loadCfg . . . 7
3.2.2 Method: updateLocalConfig . . . 7
3.2.3 Method: verifyRbfwVersion . . . 7
3.2.4 Method: bValidateMinVersion . . . 8
3.2.5 Method: bValidateMaxVersion . . . 8
3.2.6 Method: bValidateSubVersion . . . 8
3.2.7 Method: tupleVersion . . . 9
3.2.8 Method: versioncontrol_error . . . 9

4 COnFailureHandle.py **10**
4.1 Class: COnFailureHandle . . . 10
4.1.1 Method: is_noney . . . 10

5 CSetup.py **11**
5.1 Class: CSetupKeywords . . . 11
5.1.1 Method: testsuite_setup . . . 11
5.1.2 Method: testsuite_teardown . . . 11
5.1.3 Method: testcase_setup . . . 12
5.1.4 Method: testcase_teardown . . . 12
5.1.5 Method: update_config . . . 12
5.2 Class: CGeneralKeywords . . . 12
5.2.1 Method: get_config . . . 12
5.2.2 Method: load_json . . . 13

6 CStruct.py **14**
6.1 Class: CStruct . . . 14

7 Event.py **15**
7.1 Class: Event . . . 15
7.1.1 Method: trigger . . . 15

8 ScopeEvent.py **16**
8.1 Class: ScopeEvent . . . 16
8.1.1 Method: trigger . . . 16
8.2 Class: ScopeStart . . . 16
8.3 Class: ScopeEnd . . . 16

9 __init__.py **17**
9.1 Function: on . . . 17
9.2 Function: dispatch . . . 17
9.3 Function: register_event . . . 17

10 LibListener.py **18**
10.1 Class: LibListener . . . 18

11 __init__.py **19**
11.1 Class: RobotFramework_Testsuites . . . 19
11.1.1 Method: run_keyword . . . 19
11.1.2 Method: get_keyword_tags . . . 19
11.1.3 Method: get_keyword_documentation . . . 19
11.1.4 Method: failure_occurred . . . 19
11.2 Class: CTestsuitesCfg . . . 19

12 version.py **20**
12.1 Function: robfwaio_version . . . 20

13 Appendix **21**

14 History **22**

Chapter 1

Introduction

1.1 RobotFramework AIO testsuites management documentation

This is the documentation for RobotFramework_Testsuites

The RobotFramework_Testsuites package works together with JsonPreprocessor Python package to provide the enhanced features such as json configuration files, 4 different levels of configuation, global parameters, schema validation, etc.

This RobotFramework_Testsuites package will support testing for many variants of product on the same Robot project by switching between different configuration files via variant name.

Chapter 2

Description

2.1 Getting Started

2.1.1 How to install

Firstly, clone **RobotFramework_Testsuites** repository to your machine.

```
git clone
 ↪  https://github.com/test-fullautomation/robotframework-testsuitesmanagement.git
```

Go to **robotframework-testsuitesmanagement**, using the 2 common commands below to build or install this package:

```
setup.py build      will build the package underneath 'build/'
setup.py install    will install the package
```

After the build processes are completed, the package is located in **build/**, and the documents are located in **build/lib/RobotFramework_Testsuites**.

We can use `--help` to discover the options for `build` command, example:

```
setup.py build      will build the package underneath 'build/'
setup.py install    will install the package

Global options:
--verbose (-v)      run verbosely (default)
--quiet (-q)        run quietly (turns verbosity off)
--dry-run (-n)      don't actually do anything
--help (-h)         show detailed help message
--no-user-cfg       ignore pydistutils.cfg in your home directory
--command-packages  list of packages that provide distutils commands

Information display options (just display information, ignore any commands)
--help-commands     list all available commands
--name              print package name
--version (-V)      print package version
--fullname          print <package name>-<version>
--author            print the author's name
--author-email      print the author's email address
--maintainer        print the maintainer's name
--maintainer-email  print the maintainer's email address
--contact           print the maintainer's name if known, else the author's
--contact-email     print the maintainer's email address if known, else the
                    author's
--url               print the URL for this package
--license           print the license of the package
```

```
--licence           alias for --license
--description       print the package description
--long-description  print the long package description
--platforms         print the list of platforms
--classifiers       print the list of classifiers
--keywords          print the list of keywords
--provides          print the list of packages/modules provided
--requires          print the list of packages/modules required
--obsoletes         print the list of packages/modules made obsolete

usage: setup.py [global_opts] cmd1 [cmd1_opts] [cmd2 [cmd2_opts] ...]
 or: setup.py --help [cmd1 cmd2 ...]
 or: setup.py --help-commands
 or: setup.py cmd --help
```

2.2 Features

2.2.1 Using configuration files in Json format

Nowadays, Json is the leading of structuring data for exchange not only for web applications but also for other software applications. Json format is used to represent data, and become the universal standard of data exchange. That is the reason we decided using Json format for configuration files of RobotFramework AIO.

Together with `JsonPreprocessor` package, `RobotFramework_Testsuites` supports configuring RobotFramework AIO automation test project with json files which allow users to add the comments, and to import params from other json files. Adding comments and importing json files are enhanced features which are developed and documented in `JsonPreprocessor` python package.

2.2.2 Define 4 levels of configuration

`RobotFramework_Testsuites` management defines 4 different configuration levels, from level 1 to level 4. Level 1 is highest priority, and level 4 is lowest priority.

The 4 different configuration levels helps users more convenient to configure RobotFramework test project:

- Level 1 supports users execute robot run with specific configuration file.
- Level 2 supports users loading configuration file base on variant name.
- Level 3 supports users creating different separated configuration files for individual robot testsuite files.
- Level 4 supports users practicing to learn RobotFramework AIO.

Level 1: Loads configuration file via input parameter of robot command

This is highest priority of loading configuration method, that means, configuration level 2 or 3 will be ignored even it is set.

This level 1 configuration is designed for some purpose:

- In case the use wants to execute the robot run with specific configuration file for the particular purposes.
- User re-produces and verifies an issue or a corner case with new configuration file and doesn't want to modify the current configuration file.

User can address the json configuration file when executing robot testsuite with input parameter `--variable config_file:"<path_to_json_file>"`

```

```

robot --variable config_file:"<path_to_json_file>" <path_to_testsuite>

Level 2: Loads Json configuration according to variant name

This level 2 is designed for the scenario that user creates the automation testing project which running for many different variants. When trigger robot run, it will load the appropriate json configuration file.

To set RobotFramework AIO run with level 2, first user has to create a json file which contains different variants point to different configuration files.

For example, we create the `variants_cfg.json` with content below:

```
{
  "default": {
    "name": "<default_cfg_file>",
    "path": "<path>"
  },
  "variant_0": {
    "name": "<file_name_variant_0>",
    "path": "<path>"
  },
  "variant_1": {
    "name": "<file_name_variant_1>",
    "path": "<path>"
  },
  "variant_2": {
    "name": "<file_name_variant_2>",
    "path": "<path>"
  }
}
```

Then the path of `variants_cfg.json` file has to be added as input parameter of `testsuites.testsuite_setup` in `Suite Setup` of a testsuite.

In case of user wants to set configuration level 2 for entire RobotFramework test project instead of indivdiual robot testsuite file, `__init__.robot` file has to be created at the highest folder of RobotFrameowork test project, and the path of `variants_cfg.json` file has to be added as input parameter of `testsuites.testsuite_setup` in `Suite Setup` of the `__init__.robot` file.

```
*** Settings ***
Library      RobotFramework_Testsuites    WITH NAME    testsuites
Suite Setup      testsuites.testsuite_setup    <Path_to_the_file_variants_cfg.json>
```

Level 3: Find the "config/" folder in current testsuite directory

Configuration level 3 is triggered only in case of level 1 and level 2 were not set.

The configuration level 3 will check in `config/` folder in current testsuite directory the existence of json file which has the same name with testsuite file (ex: `abc.rotbot` & `./config/abc.json`), then it will load this configuration file. In case there is no json file has the same name with robot testsuite file, it will check the existence of `./config/robot_config.json` then load this `./config/robot_config.json` file as configuration file.

Level 4: Lowest priority level, it reads default configuration file

In case testsuites management library detects that configuration level 1, level 2, and level 3 are not set, the robot execution will use the configuration level 4 by default.

The default configuration file (`robot_config.json`) in installation directory:

`\RobotFramework_Testsuites\Config\robot_config.json`

2.2.3 Local configuration

In case the robot test project runs on many different test setups, each test setup has some distinguished configuration parameters. So this feature supports users create the local configuration file to override or add new parameters which are applied for indivdiual test setup.

There are 2 ways to load the local configuration for robot run:

Load local configuration via input parameter of robot command

User can address the local configuration file when executing robot testsuite with input parameter `--variable local_config:"<path_to_localconfig_file>"`

Load local configuration in default directory

To use this functionality, the `localconfig` directory has to be created with detail information below:

- **Windows:** `C:\RobotTest\localconfig`
- **Ubuntu:** `/home/<user>/RobotTest/localconfig`

User also has to create the new environment variable with name `ROBOT_LOCAL_CONFIG` and the value is the path to the `localconfig` directory.

Users can add the content to the local json configuration file `local_config.json` in the default directory above, then the configuration parameters will be overridden by the data in file `local_config.json`.

Note:

* In case loading local configuration via input parameter of robot command is using, the local configuration file `./RobotTest/localconfig/local_config.json` will be ignored.

- The value of parameters in the local configuration file do not allow nested pamameter:

  Don't allow: `"variable_need_override" : ${variable}['exist']['in_config_file']`
  Allow: "${variable}['exist']['in_config_file'] : "new value", ${variable}['new_variable'] : "value"

2.2.4 Access to configuration parameters

User can access dictionary object which is defined in configuration file in robot test script by traditional way or using ".". For example, users can call `${dict}[abc][def]` or `${dict.abc.def}`

Note: In case a parameter name contains a ".", then it is not possible to use dotdict but the traditional way `${dict}[abc][def]` is still working.

Chapter 3

CConfig.py

3.1 Class: dotdict

Imported by:

```
from RobotFramework_Testsuites.Config.CConfig import dotdict
```

Subclass: dotdict

Subclass of dict, with ”dot” (attribute) access to keys.

3.2 Class: CConfig

Imported by:

```
from RobotFramework_Testsuites.Config.CConfig import CConfig
```

Class: CConfig

Defines the properties of configuration and holds the identified config files.

The loading configuration method is divided into 4 levels, level1 is highest priority, Level4 is lowest priority.

Level1: Handed over by command line argument.

Level2: Read from content of json config file

```
{
   "default": {
      "name": "robot_config.json",
      "path": ".../config/"
   },
   "variant_0": {
      "name": "robot_config.json",
      "path": ".../config/"
   },
   "variant_1": {
      "name": "robot_config_variant_1.json",
      "path": ".../config/"
   },
      ...
      ...
}
```

According to the `ConfigName`, Testsuites-Management package will choose the corresponding config file. `"...config/"` indicats the relative path to json config file, Testsuites-Management will recursively find the `config` folder.

Level3: Read in testsuite folder `/config/robot_config.json`

Level4: Read from RobotFramework AIO install folder `/RobotFramework/defaultconfig/robot_config.json`

3.2.1 Method: loadCfg

Method: loadCfg

This loadCfg method uses to load configuration's parameters from json files.

Arguments:

- No input parameter is required

Returns:

- No return variable

3.2.2 Method: updateLocalConfig

Method: updateLocalConfig

This updateLocalConfig method updates preprocessor, global or local params base on RobotFramework AIO local config or any json config file according to purpose of specific testsuite.

Arguments:

- `sUpdateCfgFile`

  / *Condition*: required / *Type*: string
  The path of json file which wants to update configuration parameters.

Returns:

- No return variable

3.2.3 Method: verifyRbfwVersion

Method: verifyRbfwVersion

This verifyRbfwVersion validates the current RobotFramework AIO version with maximum and minimum version (if provided in the configuration file).

In case the current version is not between min and max version, then the execution of testsuite is terminated with "unknown" state

Arguments:

- No input parameter is required

Returns:

- No return variable

3.2.4 Method: bValidateMinVersion

Method: bValidateMinVersion

This bValidateMinVersion validates the current version with required minimun version.

Arguments:

- `tCurrentVersion`

  / *Condition*: required / *Type*: tuple
  Current RobotFramework AIO version.

- `tMinVersion`

  / *Condition*: required / *Type*: tuple
  The minimum version of RobotFramework AIO.

Returns:

- `True or False`

3.2.5 Method: bValidateMaxVersion

Method: bValidateMaxVersion

This bValidateMaxVersion validates the current version with required minimun version.

Arguments:

- `tCurrentVersion`

  / *Condition*: required / *Type*: tuple
  Current RobotFramework AIO version.

- `tMinVersion`

  / *Condition*: required / *Type*: tuple
  The minimum version of RobotFramework AIO.

Returns:

- `True or False`

3.2.6 Method: bValidateSubVersion

Method: bValidateSubVersion

This bValidateSubVersion validates the format of provided sub version and parse it into sub tuple for version comparision.

Arguments:

- `sVersion`

  / *Condition*: required / *Type*: string
  The version of RobotFramework AIO.

Returns:

- `lSubVersion`

  / *Type*: tuple /

3.2.7 Method: tupleVersion

Method: tupleVersion

This tupleVersion returns a tuple which contains the (major, minor, patch) version.
(remaining content needs to be fixed and restored)

Arguments:

- `sVersion`

  / *Condition*: required / *Type*: string
  The version of RobotFramework AIO.

Returns:

- `lVersion`

  / *Type*: tuple /

3.2.8 Method: versioncontrol_error

Method: versioncontrol_error

Wrapper version control error log:

Log error message of version control due to reason and set to unknown state.
reason can only be "conflict_min", "conflict_max" and "wrong_minmax".

Arguments:

- `reason`

  / *Condition*: required / *Type*: string

- `version1`

  / *Condition*: required / *Type*: string

- `version2`

  / *Condition*: required / *Type*: string

Returns:

- No return variable

Chapter 4

COnFailureHandle.py

4.1 Class: COnFailureHandle

Imported by:

```
from RobotFramework_Testsuites.Keywords.COnFailureHandle import COnFailureHandle
```

4.1.1 Method: is_noney

Chapter 5

CSetup.py

5.1 Class: CSetupKeywords

Imported by:

```
from RobotFramework_Testsuites.Keywords.CSetup import CSetupKeywords
```

Class: CSetupKeywords

This CSetupKeywords class uses to define the setup keywords which are using in suite setup and teardown of robot test script.

`Testsuite Setup` keyword loads the RobotFramework AIO configuration, checks the version of RobotFramework AIO, and logs out the basic information of the robot run.

`Testsuite Teardown` keyword currently do nothing, it's defined here for future requirements.

`Testcase Setup` keyword currently do nothing, it's defined here for future requirements.

`Testcase Teardown` keyword currently do nothing, it's defined here for future requirements.

5.1.1 Method: testsuite_setup

Method: testsuite_setup

This testsuite_setup defines the `Testsuite Setup` which is used to loads the RobotFramework AIO configuration, checks the version of RobotFramework AIO, and logs out the basic information of the robot run.

Arguments:

- `sTestsuiteCfgFile`

  / *Condition*: required / *Type*: string

  `sTestsuiteCfgFile=''` and vairiable `config_file` is not set Robotframework AIO will check for co
  level 3, and level 4.

  `sTestsuiteCfgFile` is set with a <json_config_file_path> and vairiable `config_file` is not set Robotframework AIO will load configuration level 2.

Returns:

- No return variable

5.1.2 Method: testsuite_teardown

Method: testsuite_teardown

This testsuite_teardown defines the `Testsuite Teardown` keyword, currently this keyword does nothing, it's defined here for future requirements.

5.1.3 Method: testcase_setup

Method: testcase_setup

This testcase_setup defines the `Testcase Setup` keyword, currently this keyword does nothing, it's defined here for future requirements.

5.1.4 Method: testcase_teardown

Method: testcase_teardown

This testcase_teardown defines the `Testcase Teardown` keyword, currently this keyword does nothing, it's defined here for future requirements.

5.1.5 Method: update_config

Method: update_config

This update_config defines the `Update Config` keyword which is using update the configuration object of RobotFramework AIO.

Arguments:

- `sCfgFile`

  / *Condition*: required / *Type*: string
  The path of Json configuration file.

Returns:

- No return variable

5.2 Class: CGeneralKeywords

Imported by:

```
from RobotFramework_Testsuites.Keywords.CSetup import CGeneralKeywords
```

Class: CGeneralKeywords

This CGeneralKeywords class defines the keywords which will be using in RobotFramework AIO test script.

`Get Config` keyword gets the current config object of robot run.

`Load Json` keyword loads json file then return json object.

In case new robot keyword is required, it will be defined and implemented in this class.

5.2.1 Method: get_config

Method: get_config

This get_config defines the `Get Config` keyword gets the current config object of RobotFramework AIO.

Arguments:

- No parameter is required

Returns:

- `oConfig.oConfigParams`

  / *Type*: json /

5.2.2 Method: load_json

Method: load_json

This load_json defines the `Load Json` keyword which loads json file then return json object.

Arguments:

- `jsonfile`

  / *Condition*: required / *Type*: string
  The path of Json configuration file.

- `level`

  / *Condition*: required / *Type*: int
  Level = 1 -> loads the content of jsonfile.
  level != 1 -> loads the json file which is set with variant (likes loading config level2)

Returns:

- `oJsonData`

  / *Type*: json /

Chapter 6

CStruct.py

6.1 Class: CStruct

Imported by:

```
from RobotFramework_Testsuites.Utils.CStruct import CStruct
```

Class: CStruct

This `CStruct` class creates the given attributes dynamically at runtime.

Chapter 7

Event.py

7.1 Class: Event

Imported by:

```
from RobotFramework_Testsuites.Utils.Events.Event import Event
```

7.1.1 Method: trigger

Chapter 8

ScopeEvent.py

8.1 Class: ScopeEvent

Imported by:

```
from RobotFramework_Testsuites.Utils.Events.ScopeEvent import ScopeEvent
```

8.1.1 Method: trigger

8.2 Class: ScopeStart

Imported by:

```
from RobotFramework_Testsuites.Utils.Events.ScopeEvent import ScopeStart
```

8.3 Class: ScopeEnd

Imported by:

```
from RobotFramework_Testsuites.Utils.Events.ScopeEvent import ScopeEnd
```

Chapter 9

__init__.py

9.1 Function: on

9.2 Function: dispatch

9.3 Function: register_event

Chapter 10

LibListener.py

10.1 Class: LibListener

Imported by:

```
from RobotFramework_Testsuites.Utils.LibListener import LibListener
```

Class: LibListener

This `LibListener` class defines the hook methods.

- `_start_suite` hooks to every starting testsuite of robot run.
- `_end_suite` hooks to every ending testsuite of robot run.
- `_start_test` hooks to every starting test case of robot run.
- `_end_test` hooks to every ending test case of robot run.

Chapter 11

__init__.py

11.1 Class: RobotFramework_Testsuites

Imported by:

```
from RobotFramework_Testsuites.__init__ import RobotFramework_Testsuites
```

Class: RobotFramework_Testsuites

RobotFramework_Testsuites is the Bosch testing library for Robot Framework.
RobotFramework_Testsuites control peripheral devices, tools and target under testing.

11.1.1 Method: run_keyword

11.1.2 Method: get_keyword_tags

11.1.3 Method: get_keyword_documentation

11.1.4 Method: failure_occurred

11.2 Class: CTestsuitesCfg

Imported by:

```
from RobotFramework_Testsuites.__init__ import CTestsuitesCfg
```

Chapter 12

version.py

12.1 Function: robfwaio_version

Return testsuitemanagement version as Robot framework AIO version

Chapter 13

Appendix

About this package:

Table 13.1: Package setup

Setup parameter	Value
Name	RobotFramework_Testsuites
Version	0.2.3
Date	30.11.2022
Description	Functionality to manage RobotFramework testsuites
Package URL	robotframework-testsuitesmanagement
Author	Mai Dinh Nam Son
Email	son.maidinhnam@vn.bosch.com
Language	Programming Language :: Python :: 3
License	License :: OSI Approved :: Apache Software License
OS	Operating System :: OS Independent
Python required	>=3.0
Development status	Development Status :: 4 - Beta
Intended audience	Intended Audience :: Developers
Topic	Topic :: Software Development

Chapter 14

History

0.1.0	06/2022
Initial version	
0.2.3	11/2022
Added local configuration feature	

RobotFramework_Testsuites.pdf
Created at 06.12.2022 - 18:09:34
by GenPackageDoc v. 0.32.0