

TestResultDBAccess

v. 0.1.0

Tran Duy Ngoan

25.04.2024

Contents

1	Introduction	1
2	Description	2
2.1	Get Robot Framework XML result	2
2.1.1	Robot Framework test case Settings	2
2.1.2	Sample Robot Framework Test Case	3
2.1.3	Execute Robot Framework test case(s) to get result file	4
2.2	Tool features	5
2.2.1	Usage	5
2.2.2	Verify provided arguments	6
2.2.3	Searching *.xml result file(s)	6
2.2.4	Handle essential information for TestResultWebApp	6
2.2.5	Append to existing execution result	8
2.2.6	Switch Database Access Interface	9
2.3	Display on WebApp	10
3	DBAccessInterface.py	11
3.1	Class: DBAccess	11
3.1.1	Method: connect	12
3.1.2	Method: disconnect	12
3.1.3	Method: commit	12
3.1.4	Method: arGetCategories	12
3.1.5	Method: bExistingResultID	12
3.1.6	Method: sGetLatestFileID	12
3.1.7	Method: sCreateNewTestResult	12
3.1.8	Method: nCreateNewFile	12
3.1.9	Method: vCreateNewHeader	12
3.1.10	Method: nCreateNewSingleTestCase	12
3.1.11	Method: nCreateNewTestCase	12
3.1.12	Method: vCreateAbortReason	12
3.1.13	Method: vCreateCCRdata	12
3.1.14	Method: vCreateTags	12
3.1.15	Method: vCreateReanimation	12
3.1.16	Method: vSetCategory	12
3.1.17	Method: vUpdateFileEndTime	12
3.1.18	Method: vUpdateResultEndTime	12
3.1.19	Method: vUpdateEvtbl	12

3.1.20	Method: vUpdateEvtbls	12
3.1.21	Method: vFinishTestResult	12
4	DirectDBAccess.py	13
4.1	Class: DirectDBAccess	13
4.1.1	Method: connect	13
4.1.2	Method: commit	14
4.1.3	Method: disconnect	14
4.1.4	Method: cleanAllTables	14
4.1.5	Method: sCreateNewTestResult	14
4.1.6	Method: nCreateNewFile	15
4.1.7	Method: vCreateNewHeader	16
4.1.8	Method: nCreateNewSingleTestCase	18
4.1.9	Method: nCreateNewTestCase	19
4.1.10	Method: vCreateTags	21
4.1.11	Method: vSetCategory	21
4.1.12	Method: vUpdateStartEndTime	21
4.1.13	Method: arGetCategories	22
4.1.14	Method: vCreateAbortReason	22
4.1.15	Method: vCreateReanimation	22
4.1.16	Method: vCreateCCRdata	22
4.1.17	Method: vFinishTestResult	23
4.1.18	Method: vUpdateEvtbls	23
4.1.19	Method: vUpdateEvtbl	23
4.1.20	Method: vEnableForeignKeyCheck	23
4.1.21	Method: sGetLatestFileID	24
4.1.22	Method: vUpdateFileEndTime	24
4.1.23	Method: vUpdateResultEndTime	24
4.1.24	Method: bExistingResultID	24
4.1.25	Method: arGetProjectVersionSWByID	25
5	RestApiDBAccess.py	26
5.1	Class: RestApiDBAccess	26
5.1.1	Method: get_certs_file	27
5.1.2	Method: encrypt_password	27
5.1.3	Method: connect	27
5.1.4	Method: disconnect	27
5.1.5	Method: arGetCategories	27
5.1.6	Method: bExistingResultID	27
5.1.7	Method: sGetLatestFileID	27
5.1.8	Method: arGetProjectVersionSWByID	27
5.1.9	Method: sCreateNewTestResult	27
5.1.10	Method: nCreateNewFile	27
5.1.11	Method: vCreateNewHeader	27
5.1.12	Method: nCreateNewSingleTestCase	27
5.1.13	Method: nCreateNewTestCase	27
5.1.14	Method: vCreateAbortReason	27

5.1.15	Method: vCreateCCRdata	27
5.1.16	Method: vCreateTags	27
5.1.17	Method: vCreateReanimation	27
5.1.18	Method: vSetCategory	27
5.1.19	Method: vUpdateFileEndTime	27
5.1.20	Method: vUpdateResultEndTime	27
5.1.21	Method: vFinishTestResult	27
5.1.22	Method: vUpdateEvtbl	27
5.1.23	Method: vUpdateEvtbls	27
6	DBAccessFactory.py	28
6.1	Class: DBAccessFactory	28
6.1.1	Method: create	28
7	Appendix	29
8	History	30

Chapter 1

Introduction

TestResultDBAccess is a command-line tool that enables you to import Robot Framework XML result files into TestResultWebApp's database for presenting an overview about the whole test execution and details of each test result.

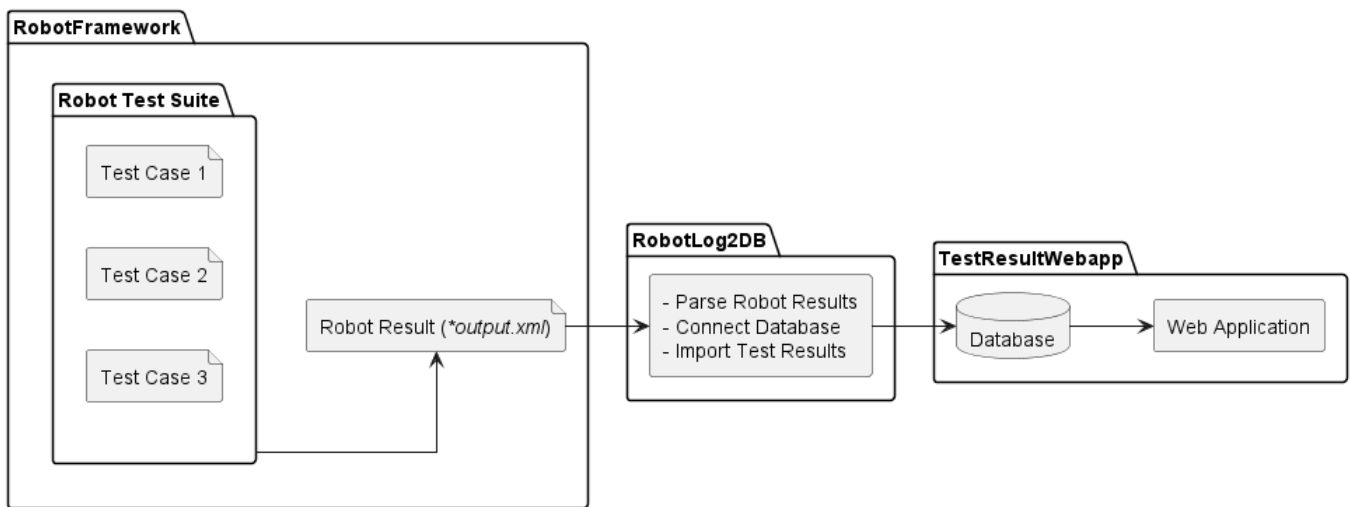


Figure 1.1: Tool data flow

TestResultDBAccess tool requires several arguments, including the location of the [Robot Framework XML result file\(s\)](#) to parse all information of test execution result and TestResultWebApp's database credential for importing that result.

[TestResultWebApp](#) requires some mandatory information to manage and display the test result properly. Therefore, they should be provided in Robot Framework test case before execution, then they will be available in XML result file for importing.

However, you can use optional arguments of **TestResultDBAccess** tool to provide those data if they are missing or you want to overwrite them with the expected values.

Finally, **TestResultDBAccess** also allows you to append existing results in the database, which is helpful when you need to update previous test results or add the missing XML result file(s) from previous tool execution.

Chapter 2

Description

TestResultWebApp requires **project/variant**, **software version** as essential data to manage results in database and some optional information such as **component**, ... to visualize test data properly on web application.

These data are not the mandatory fields of Robot Framework test case or XML result file. So that, they should be defined as `Metadata` and `[Tags]` in Robot Framework test case, then the generated XML result file includes all necessary information of test execution for importing to TestResultWebApp's database.

The next section [Get Robot Framework XML result](#) will introduce you the Robot Framework test case settings to make necessary information available for displaying on TestResultWebApp.

In case you have already had the Robot Framework XML file(s) without those definitions, do not worry because **TestResultDBAccess** will handle those data with default values.

Futhermore, you can specify them with your expected values by providing as command line arguments when executing **TestResultDBAccess** tool. The detail of those command line arguments are described in [Specify essential information with optional arguments](#) section.

2.1 Get Robot Framework XML result

2.1.1 Robot Framework test case Settings

The below document is recommended Robot Framework test case `Settings` before execution. So that, the generated `*.xml` result file will contain all necessary information.

For the whole test execution:

- Project/Variant

```
Metadata    project    ${Project_name}
```

- Versions

```
Metadata    version_hw    ${Software_version}
Metadata    version_hw    ${Hardware_version}
Metadata    version_test  ${Test_version}
```

For the Suite/File information:

- Description/Documentation

```
Documentation  ${Suite_description}
```

- Author

```
Metadata    author    ${Author_name}
```

- Component

```
Metadata    component    ${Component_name}
```

- Test Tool - Test framework and Python version, e.g **Robot Framework 3.2rc2 (Python 3.9.0 on win32)**

```
Metadata    testtool    ${Test_tool}
```

- Test Machine

```
Metadata    machine    ${COMPUTERNAME}
```

- Tester

```
Metadata    tester    ${USER}
```

For test case information:

- Issue ID

```
[Tags]      ISSUE-${ISSUE_ID}
```

- Testcase ID

```
[Tags]      TCID-${TC_ID}
```

- Requirement ID

```
[Tags]      FID-${REQ_ID}
```

2.1.2 Sample Robot Framework Test Case

Sample Robot Framework test case with the necessary information for importing to TestResultWebApp's database:

```
*** Settings ***
# Test execution level
Metadata    project      ROBFW          # Project/Variant
Metadata    version_sw   SW_VERSION_0.1  # Software version
Metadata    version_hw   HW_VERSION_0.1  # Hardware version
Metadata    version_test TEST_VERSION_0.1 # Test version

# File/Suite level
Documentation      This is description for robot test file
Metadata    author      Tran Duy Ngoan (RBVH/ECM1)
Metadata    component    Import_Tools
Metadata    testtool     Robot Framework 4.1.3 (Python 3.9.16 on win32)
Metadata    machine      ${COMPUTERNAME}
Metadata    tester       ${USER}

*** Test Cases ***
Testcase 01
    [Tags]    ISSUE-001    TCID-1001    FID-112    FID-111
    Log       This is Testcase 01

Testcase 02
    [Tags]    ISSUE-RTC-003    TCID-1002    FID-113
    Log       This is Testcase 01
```

Listing 2.1: Sample Robot Framework testcase

Hint

Instead of setting `Metadata` in Robot Framework test case, you can also specify them as the `--metadata` [command line argument](#) when executing Robot Framework test cases. Furthermore, in case you are using RobotFramework AIO, above highlighted `Metadata` definitions are not required because they have been handled by `RobotFramework_TestsuitesManagement` library within `Suite Setup` .

2.1.3 Execute Robot Framework test case(s) to get result file

Now, execute your Robot Framework test case(s) with your IDE or [using command line](#) to get the Robot result file

```
robot your_testcases.robot
```

Or with python module

```
python -m robot your_testcases.robot
```

The default name of Robot result file is `output.xml`. Its filename can be changed by specifying other filename with argument `--output (-o)` when executing Robot Framework test case(s)

```
robot your_testcases.robot --output path/to/robot_result.xml
```


2.2 Tool features

After getting the Robot Framework **.xml* result file(s), you can use the **TestResultDBAccess** tool to import the test results into TestResultWebApp's database.

Its usage and enhance features are described as below.

2.2.1 Usage

Use below command to get tools's usage:

```
TestResultDBAccess -h
```

The tool's usage should be showed as below:

```
usage: TestResultDBAccess (RobotXMLResult to TestResultWebApp importer) [-h] [-v]
      [--recursive] [--dryrun] [--append] [--UUID UUID]
      [--variant VARIANT] [--versions VERSIONS] [--config CONFIG]
      resultxmlfile server user password database

TestResultDBAccess imports XML result files (default: output.xml) generated by the
Robot Framework into a WebApp database.

positional arguments:
resultxmlfile          absolute or relative path to the result file or directory
                        of result files to be imported.
server                 server which hosts the database (IP or URL).
user                   user for database login.
password               password for database login.
database               database schema for database login.

optional arguments:
-h, --help             show this help message and exit
-v, --version          version of the TestResultDBAccess importer.
--recursive            if set, then the path is searched recursively for output
                        files to be imported.
--dryrun               if set, then verify all input arguments (includes DB
                        connection) and show what would be done.
--append               is used in combination with --UUID UUID.If set, allow to
                        append new result(s) to existing execution result UUID in
                        --UUID argument.
--UUID UUID            UUID used to identify the import and version ID on webapp.
                        If not provided TestResultDBAccess will generate an UUID for the
                        whole import.
--variant VARIANT      variant name to be set for this import.
--versions VERSIONS    metadata: Versions (Software;Hardware;Test) to be set for
                        this import (semicolon separated).
--config CONFIG        configuration json file for component mapping information.
--interface {db,rest}  database access interface.
```

As above instruction, **TestResultDBAccess** tool requires 5 positional arguments which contains all required information for importing.

The below command is simple usage with all required arguments to import robot results into TestResultWebApp's database:

```
TestResultDBAccess output.xml localhost db.user db_pw db_name
```

Besides the executable file **TestResultDBAccess** you can also run tool as a Python module

```
python -m TestResultDBAccess output.xml localhost db.user db_pw db_name
```

Beside the required arguments, there are optional arguments which provides some enhance features as the following sections.

2.2.2 Verify provided arguments

Sometimes, we just want to validate the **.xml* and database connection without changing anything in the database, the optional argument `--dryrun` can be used in this case.

When executing in dryrun mode, **TestResultDBAccess** will:

- Verify the provided Robot Framework **.xml* result file is valid or not.
- Verify the database connection with provided credential.
- Verify other information which given in optional arguments.
- Just print all test cases will be imported without touching database.

This feature will helps you to ensure that there is no error when executing **TestResultDBAccess** tool (normal mode) to create new record(s) and update TestResultWebApp's database.

2.2.3 Searching *.xml result file(s)

The first argument `resultxmlfile` of **TestResultDBAccess** can be a single file or the folder that contains multiple result files.

When the folder is used, **TestResultDBAccess** will only search for **.xml* file(s) under given directory and exclude any file within subdirectories as default.

In case you have result file(s) under the subdirectory of given folder and want these result files will also be imported, the optional argument `--recursive` should be used when executing **TestResultDBAccess** command.

When `--recursive` argument is set, **TestResultDBAccess** will walk through the given directory and its subdirectories to discover and collect all available **.xml* for importing.

For example: your result folder has a structure as below:

```
logFolder
|_____ result_1.xml
|_____ result_2.xml
|_____ subFolder_1
|           |_____ result_sub_1.xml
|           |_____ subSubFolder
|                   |_____ result_sub_sub_1.xml
|_____ subFolder_2
|           |_____ result_sub_2.xml
```

- Without `--recursive` : only **result_1.xml** and **result_2.xml** are found for importing.
- With `--recursive` : all **result_1.xml**, **result_2.xml**, **result_sub_1.xml**, **result_sub_2.xml** and **result_sub_sub_1.xml** will be imported.

2.2.4 Handle essential information for TestResultWebApp

Default values

TestResultWebApp requires **Project**, **Software version** to manage the execution results and **Component** to group test cases in the displayed charts.

In case above information is missing in [test case settings](#) during the test case execution, that leads to the missing information in the *output.xml* result file. So, this missing information will be set to default values when importing with **TestResultDBAccess** tool:

- **Project**: will be set to default value `ROBFW` if not defined.
- **Software version**: will be set to execution time `%Y%m%d_%H%M%S` as default value.
- **Component**: will be set to default value `unknown` if not defined.

Specify essential information with optional arguments

You can also provide essential information in command line when executing the [TestResultDBAccess](#) tool with below optional arguments:

Warning



These below settings may overwrite the existing information of `project`, `version_sw` and `component` which have been defined in Robot Framework test case in [above section](#). So, **only** use these arguments in case you want to define the missing information or overwrite the existing values with your expected values for the importing result.

- `--variant VARIANT` To specify the **Project/Variant** information.
- `--versions VERSIONS` To specify the **Software, Hardware** and **Test** versions information.
- `--config CONFIG` To provide a configuration `*.json` file as `CONFIG` argument. Currently, the configuration `*.json` supports below settings:
 - `"variant"` to specify the **Project/Variant** as `string` value.
 - `"version_sw"` to specify the **Software version** information as `string` value.
 - `"version_hw"` to specify the **Hardware under-test version** as `string` value.
 - `"version_test"` to specify the **Test version** as `string` value.

Notice



These above settings with `--config CONFIG` will have lower priority than the commandline arguments `--variant VARIANT` and `--versions VERSIONS`

- `"testtool"` to specify the **Test toolchain** as `string` value.
- `"tester"` to specify the **Test user** as `string` value.
- `"components"` to specify the **Component** information which will be displayed on [TestResultWebApp](#). Value can be:

- * `string` : to specify the same **Component** for all test casea within this execution.

```
{
  "components" : "atest",
  ...
}
```

- * `object` : to define the mapping json object between **Component** info as key and a directory path (or list of directory paths) to Robot Framework test file (`*.robot` file) as value.

```
{
  "components" : {
    "cli"       : "robot/cli",
    "core"      : "robot/core",
    "external"  : "robot/external",
    "keywords"  : "robot/keywords",
    "libdoc"    : "robot/libdoc",
    "connectivity" : [
      "selftest/serial",
      "selftest/ssh",
      "selftest/tcpip"
    ]
  },
  ...
}
```

As above sample configuration, the `"components"` key contains the mappings for individual components, such as **cli**, **core**, **external**, **keywords** and **libdoc**, where the value is the directory path for the corresponding Robot Framework files.

Additionally, the **connectivity** key is an example of a mapping where the value is a list of directory paths, indicating that the **connectivity** component is composed of all Robot Framework files located in multiple directories, namely **selftest/serial**, **selftest/ssh** and **selftest/tcpip**.

In other words, the component mapping can be explained as below:

- Test case(s) under **robot/cli** folder is belong **cli** component
- Test case(s) under **robot/core** folder is belong **core** component
- ...
- **connectivity** component contains all test case(s) which is located under **selftest/serial**, **selftest/ssh** and **selftest/tcpip** folders.

In case the given configuration **.json* is not valid or unsupported key is used, the corresponding error will be raised.

2.2.5 Append to existing execution result

TestResultDBAccess also allows you to append new test result(s) (missing from previous import, on other test setup, ...) into the existing execution result (identified by the **UUID**) in TestResultWebApp's database. The combination of optional arguments `--UUID <UUID>` and `--append` should be used in this case.

The `--append` makes **TestResultDBAccess** run in append mode and the `--UUID <UUID>` is used to specify the existing UUID of execution result to be appended.

For example, the result with UUID **c7991c07-4de2-4d65-8568-00c5556c82aa** is already existing in TestResultWebApp's database and you want to append result(s) in **output.xml** into that execution result.

The command will be used as below:

```
python -m TestResultDBAccess output.xml localhost testuser testpw testdb --UUID ↩
↪ c7991c07-4de2-4d65-8568-00c5556c82aa --append
```

If the argument `--UUID <UUID>` is used without `--append` :

- An error will be thrown and the import job is terminated immediately if the provided **UUID** is already existing

```
FATAL ERROR: Execution result with UUID 'c7991c07-4de2-4d65-8568-00c5556c82aa' is ↩
↪ already existing.
Please use other UUID (or remove '--UUID' argument from your command) ↩
↪ for new execution result.
Or add '--append' argument in your command to append new result(s) to ↩
↪ this existing UUID.
```

- The importing execution result will have an identifier as the provided **UUID** if that **UUID** is not existing

If the argument `--append` is used and:

- given UUID in `--UUID <UUID>` argument is existing: the new result(s) will be appended to that UUID
- given UUID in `--UUID <UUID>` argument is not existing: tool will be terminated immediately with below error

```
FATAL ERROR: Execution result with UUID 'c7991c07-4de2-4d65-8568-00c5556c82aa' is ↩
↪ not existing for appending.
Please use an existing UUID to append new result(s) to that UUID.
Or remove '--append' argument in your command to create new execution ↩
↪ result with given UUID.
```

- `--UUID <UUID>` is not provided: tool will be terminated immediately with below error

```
FATAL ERROR: '--append' argument should be used in combination with '--UUID UUID' ↩
↪ argument
```

Notice

When using append mode and `project` / `variant` , `version_sw` are provided within `--variant VARIANT` , `--versions VERSIONS` or `--config CONFIG` arguments, they will be validated with the existing values in database.

An error will be raised in case the given value is not matched with the existing ones. E.g:

```
FATAL ERROR: Given version software 'my_version' is different with existing ↵  
↵ value 'SW01' in database.
```

2.2.6 Switch Database Access Interface

2.3 Display on WebApp

As soon as the *output.xml* file(s) is imported successfully to database, the result for that execution will be available on [TestResultWebApp](#).

Above test case settings in Robot Framework test case will be reflected on **Dashboard** (General view) and **Data table** (Detailed view) as below figures:

Execution result metadata:

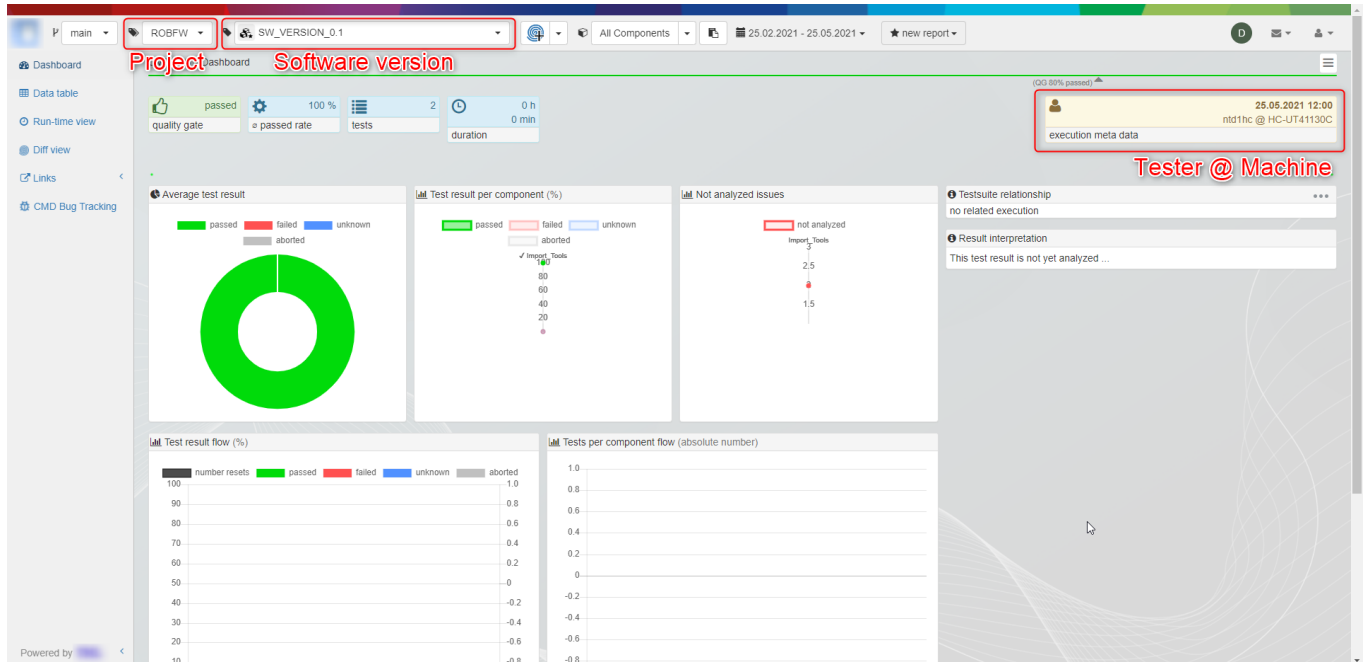


Figure 2.1: Dashboard view

Suite/File metadata and test case information:

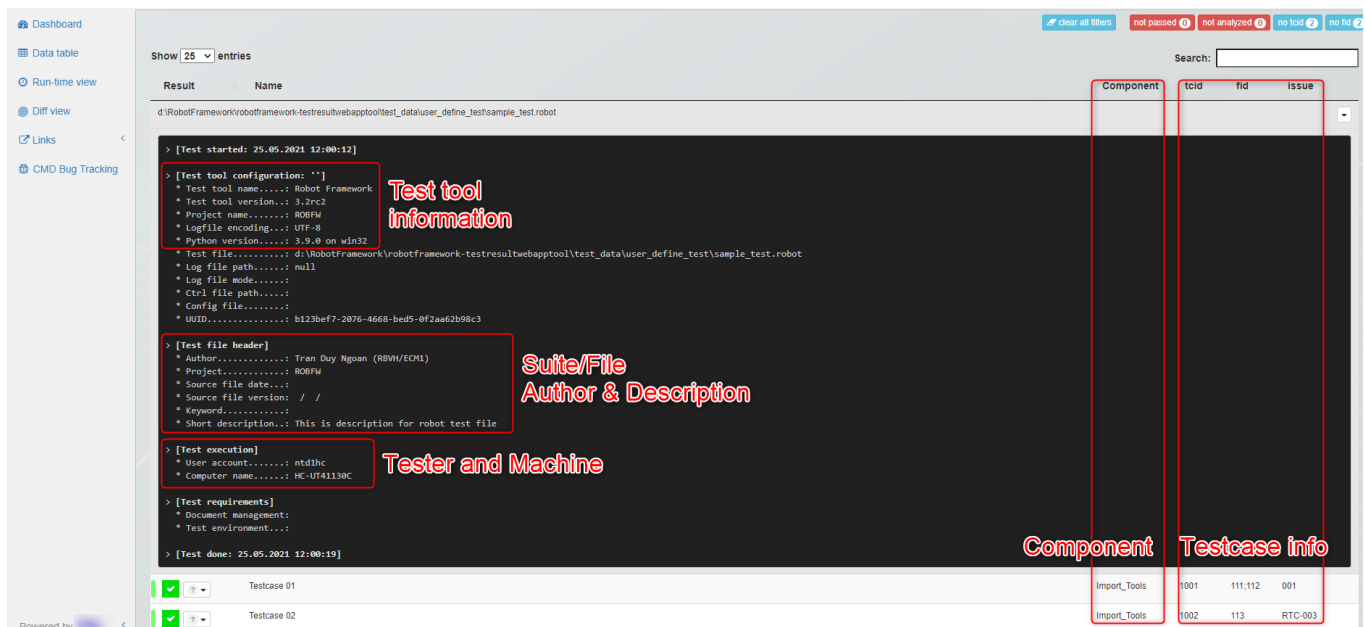


Figure 2.2: Datatable view

Chapter 3

DBAccessInterface.py

3.1 Class: DBAccess

Imported by:

```
from TestResultDBAccess.DBAccess.DBAccessInterface import DBAccess
```

- 3.1.1 Method: connect
- 3.1.2 Method: disconnect
- 3.1.3 Method: commit
- 3.1.4 Method: arGetCategories
- 3.1.5 Method: bExistingResultID
- 3.1.6 Method: sGetLatestFileID
- 3.1.7 Method: sCreateNewTestResult
- 3.1.8 Method: nCreateNewFile
- 3.1.9 Method: vCreateNewHeader
- 3.1.10 Method: nCreateNewSingleTestCase
- 3.1.11 Method: nCreateNewTestCase
- 3.1.12 Method: vCreateAbortReason
- 3.1.13 Method: vCreateCCRdata
- 3.1.14 Method: vCreateTags
- 3.1.15 Method: vCreateReanimation
- 3.1.16 Method: vSetCategory
- 3.1.17 Method: vUpdateFileEndTime
- 3.1.18 Method: vUpdateResultEndTime
- 3.1.19 Method: vUpdateEvtbl
- 3.1.20 Method: vUpdateEvtbls
- 3.1.21 Method: vFinishTestResult

Chapter 4

DirectDBAccess.py

4.1 Class: DirectDBAccess

Imported by:

```
from TestResultDBAccess.DBAccess.DirectDBAccess import DirectDBAccess
```

DirectDBAccess class play a role as mysqlclient and provide methods to interact with TestResultWebApp's database.

4.1.1 Method: connect

Connect to the database with provided authentication and db info.

Arguments:

- `host`
/ *Condition*: required / *Type*: str /
URL which is hosted the TestResultWebApp's database.
- `user`
/ *Condition*: required / *Type*: str /
User name for database authentication.
- `passwd`
/ *Condition*: required / *Type*: str /
User's password for database authentication.
- `database`
/ *Condition*: required / *Type*: str /
Database name.
- `charset`
/ *Condition*: optional / *Type*: str / *Default*: 'utf8' /
The connection character set.
- `use_unicode`
/ *Condition*: optional / *Type*: bool / *Default*: True /
If True, CHAR and VARCHAR and TEXT columns are returned as Unicode strings, using the configured character set.

Returns:

(no returns)

4.1.2 Method: commit

Commit changes within transaction.

Arguments:

(no arguments)

Returns:

(no returns)

4.1.3 Method: disconnect

Disconnect from TestResultWebApp's database.

Arguments:

(no arguments)

Returns:

(no returns)

4.1.4 Method: cleanAllTables

Delete all table data. Please be careful before calling this method.

Arguments:

(no arguments)

Returns:

(no returns)

4.1.5 Method: sCreateNewTestResult

Creates a new test result in `tbl_result`. This is the main table which is linked to all other data by means of `test_result_id`.

Arguments:

- `tbl_prj_project`
/ Condition: required / Type: str /
Project information.
- `tbl_prj_variant`
/ Condition: required / Type: str /
Variant information.
- `tbl_prj_branch`
/ Condition: required / Type: str /
Branch information.
- `tbl_test_result_id`
/ Condition: required / Type: str /
UUID of test result.
- `tbl_result_interpretation`
/ Condition: required / Type: str /
Result interpretation.
- `tbl_result_time_start`
/ Condition: required / Type: str /
Test result start time as format `%Y-%m-%d %H:%M:%S`.

- `.tbl_result_time_end`
/ *Condition*: required / *Type*: str /
Test result end time as format %Y-%m-%d %H:%M:%S.
- `.tbl_result_version_sw_target`
/ *Condition*: required / *Type*: str /
Software version information.
- `.tbl_result_version_sw_test`
/ *Condition*: required / *Type*: str /
Test version information.
- `.tbl_result_version_target`
/ *Condition*: required / *Type*: str /
Hardware version information.
- `.tbl_result_jenkinsurl`
/ *Condition*: required / *Type*: str /
Jenkinsurl in case test result is executed by jenkins.
- `.tbl_result_reporting_qualitygate`
/ *Condition*: required / *Type*: str /
Qualitygate information for reporting.

Returns:

- `.tbl_test_result_id`
/ *Type*: str /
`test_result_id` of new test result.

4.1.6 Method: nCreateNewFile

Create new file entry in `tbl_file` table.

Arguments:

- `.tbl_file_name`
/ *Condition*: required / *Type*: str /
File name information.
- `.tbl_file_tester_account`
/ *Condition*: required / *Type*: str /
Tester account information.
- `.tbl_file_tester_machine`
/ *Condition*: required / *Type*: str /
Test machine information.
- `.tbl_file_time_start`
/ *Condition*: required / *Type*: str /
Test file start time as format %Y-%m-%d %H:%M:%S.
- `.tbl_file_time_end`
/ *Condition*: required / *Type*: str /
Test file end time as format %Y-%m-%d %H:%M:%S.

- `_tbl_test_result_id`
/ *Condition*: required / *Type*: str /
UUID of test result for linking to `tbl_result` table.
- `_tbl_file_origin`
/ *Condition*: required / *Type*: str /
Origin (test framework) of test file. Default is "ROBFW"

Returns:

- `iInsertedID`
/ *Type*: int /
ID of new entry.

4.1.7 Method: vCreateNewHeader

Create a new header entry in `tbl_file.header` table which is linked with the file.

Arguments:

- `_tbl_file_id`
/ *Condition*: required / *Type*: int /
File ID information.
- `_tbl_header_testtoolconfiguration_testtoolname`
/ *Condition*: required / *Type*: str /
Test tool name.
- `_tbl_header_testtoolconfiguration_testtoolversionstring`
/ *Condition*: required / *Type*: str /
Test tool version.
- `_tbl_header_testtoolconfiguration_projectname`
/ *Condition*: required / *Type*: str /
Project name.
- `_tbl_header_testtoolconfiguration_logfileencoding`
/ *Condition*: required / *Type*: str /
Encoding of logfile.
- `_tbl_header_testtoolconfiguration_pythonversion`
/ *Condition*: required / *Type*: str /
Python version info.
- `_tbl_header_testtoolconfiguration_testfile`
/ *Condition*: required / *Type*: str /
Test file name.
- `_tbl_header_testtoolconfiguration_logfilepath`
/ *Condition*: required / *Type*: str /
Path to log file.
- `_tbl_header_testtoolconfiguration_logfilemode`
/ *Condition*: required / *Type*: str /
Mode of log file.

- `.tbl.header.testtoolconfiguration.ctrlfilepath`
/ *Condition*: required / *Type*: str /
Path to control file.
- `.tbl.header.testtoolconfiguration.configfile`
/ *Condition*: required / *Type*: str /
Path to configuration file.
- `.tbl.header.testtoolconfiguration.confname`
/ *Condition*: required / *Type*: str /
Configuration name.
- `.tbl.header.testfileheader.author`
/ *Condition*: required / *Type*: str /
File author.
- `.tbl.header.testfileheader.project`
/ *Condition*: required / *Type*: str /
Project information.
- `.tbl.header.testfileheader.testfiledate`
/ *Condition*: required / *Type*: str /
File creation date.
- `.tbl.header.testfileheader.version.major`
/ *Condition*: required / *Type*: str /
File major version.
- `.tbl.header.testfileheader.version.minor`
/ *Condition*: required / *Type*: str /
File minor version.
- `.tbl.header.testfileheader.version.patch`
/ *Condition*: required / *Type*: str /
File patch version.
- `.tbl.header.testfileheader.keyword`
/ *Condition*: required / *Type*: str /
File keyword.
- `.tbl.header.testfileheader.shortdescription`
/ *Condition*: required / *Type*: str /
File short description.
- `.tbl.header.testexecution.useraccount`
/ *Condition*: required / *Type*: str /
Tester account who run the execution.
- `.tbl.header.testexecution.computername`
/ *Condition*: required / *Type*: str /
Machine name which is executed on.
- `.tbl.header.testrequirements.documentmanagement`
/ *Condition*: required / *Type*: str /
Requirement management information.

- `tbl_header.testrequirements.testenvironment`
/ *Condition*: required / *Type*: str /
Requirement environment information.
- `tbl_header.testbenchconfig_name`
/ *Condition*: required / *Type*: str /
Testbench configuration name.
- `tbl_header.testbenchconfig_data`
/ *Condition*: required / *Type*: str /
Testbench configuration data.
- `tbl_header.preprocessor_filter`
/ *Condition*: required / *Type*: str /
Preprocessor filter information.
- `tbl_header.preprocessor_parameters`
/ *Condition*: required / *Type*: str /
Preprocessor parameters definition.

Returns:*(no returns)***4.1.8 Method: nCreateNewSingleTestCase**Create single testcase entry in `tbl_case` table immediately.**Arguments:**

- `tbl_case_name`
/ *Condition*: required / *Type*: str /
Test case name.
- `tbl_case_issue`
/ *Condition*: required / *Type*: str /
Test case issue ID.
- `tbl_case_tcid`
/ *Condition*: required / *Type*: str /
Test case ID (used for testmanagement tool).
- `tbl_case_fid`
/ *Condition*: required / *Type*: str /
Test case requirement (function) ID.
- `tbl_case_testnumber`
/ *Condition*: required / *Type*: int /
Order of test case in file.
- `tbl_case_repeatcount`
/ *Condition*: required / *Type*: int /
Test case repeatition count.
- `tbl_case_component`
/ *Condition*: required / *Type*: str /
Component which test case is belong to.

- `tbl.case.time_start`
/ *Condition*: required / *Type*: str /
Test case start time as format %Y-%m-%d %H:%M:%S.
- `tbl.case.result_main`
/ *Condition*: required / *Type*: str /
Test case main result.
- `tbl.case.result_state`
/ *Condition*: required / *Type*: str /
Test case completion state.
- `tbl.case.result_return`
/ *Condition*: required / *Type*: int /
Test case result code (as integer).
- `tbl.case.counter_resets`
/ *Condition*: required / *Type*: int /
Counter of target reset within test case execution.
- `tbl.case.lastlog`
/ *Condition*: required / *Type*: str /
Traceback information when test case is failed.
- `tbl.test.result_id`
/ *Condition*: required / *Type*: str /
UUID of test result for linking to file in `tbl.result` table.
- `tbl.file.id`
/ *Condition*: required / *Type*: int /
Test file ID for linking to file in `tbl.file` table.

Returns:

- `iInsertedID`
/ *Type*: int /
ID of new entry.

4.1.9 Method: `nCreateNewTestCase`

Create bulk of test case entries: new test cases are buffered and inserted as bulk.

Once `_NUM_BUFFERD_ELEMENTS_FOR_EXECUTEMANY` is reached, the creation query is executed.

Arguments:

- `tbl.case.name`
/ *Condition*: required / *Type*: str /
Test case name.
- `tbl.case.issue`
/ *Condition*: required / *Type*: str /
Test case issue ID.
- `tbl.case.tcid`
/ *Condition*: required / *Type*: str /
Test case ID (used for testmanagement tool).

- `tbl.case.fid`
/ *Condition*: required / *Type*: str /
Test case requirement (function) ID.
- `tbl.case.testnumber`
/ *Condition*: required / *Type*: int /
Order of test case in file.
- `tbl.case.repeatcount`
/ *Condition*: required / *Type*: int /
Test case repetition count.
- `tbl.case.component`
/ *Condition*: required / *Type*: str /
Component which test case is belong to.
- `tbl.case.time_start`
/ *Condition*: required / *Type*: str /
Test case start time as format %Y-%m-%d %H:%M:%S.
- `tbl.case.result_main`
/ *Condition*: required / *Type*: str /
Test case main result.
- `tbl.case.result_state`
/ *Condition*: required / *Type*: str /
Test case completion state.
- `tbl.case.result_return`
/ *Condition*: required / *Type*: int /
Test case result code (as integer).
- `tbl.case.counter_resets`
/ *Condition*: required / *Type*: int /
Counter of target reset within test case execution.
- `tbl.case.lastlog`
/ *Condition*: required / *Type*: str /
Traceback information when test case is failed.
- `tbl.test.result_id`
/ *Condition*: required / *Type*: str /
UUID of test result for linking to file in `tbl.result` table.
- `tbl.file_id`
/ *Condition*: required / *Type*: int /
Test file ID for linking to file in `tbl.file` table.

Returns:*(no returns)*

4.1.10 Method: vCreateTags

Create tag entries.

Arguments:

- `.tbl_test_result_id`
/ *Condition*: required / *Type*: str /
UUID of test result.
- `.tbl_usr_result_tags`
/ *Condition*: required / *Type*: str /
User tags information.

Returns:

(no returns)

4.1.11 Method: vSetCategory

Create category entry.

Arguments:

- `.tbl_test_result_id`
/ *Condition*: required / *Type*: str /
UUID of test result.
- `tbl_result_category_main`
/ *Condition*: required / *Type*: str /
Category information.

Returns:

(no returns)

4.1.12 Method: vUpdateStartTimeEndTime

Create start-end time entry.

Arguments:

- `.tbl_test_result_id`
/ *Condition*: required / *Type*: str /
UUID of test result.
- `.tbl_result_time_start`
/ *Condition*: required / *Type*: str /
Result start time as format %Y-%m-%d %H:%M:%S.
- `.tbl_result_time_end`
/ *Condition*: required / *Type*: str /
Result end time as format %Y-%m-%d %H:%M:%S.

Returns:

(no returns)

4.1.13 Method: arGetCategories

Get existing categories.

Arguments:

(no arguments)

Returns:

- `arCategories`
/ *Type*: list /
List of existing categories.

4.1.14 Method: vCreateAbortReason

Create abort reason entry.

Arguments:

- `_tbl_test_result_id`
/ *Condition*: required / *Type*: str /
UUID of test result.
- `_tbl_abort_reason`
/ *Condition*: required / *Type*: str /
Abort reason.
- `_tbl_abort_message`
/ *Condition*: required / *Type*: str /
Detail message of abort.

Returns:

(no returns)

4.1.15 Method: vCreateReanimation

Create reanimation entry.

Arguments:

- `_tbl_test_result_id`
/ *Condition*: required / *Type*: str /
UUID of test result.
- `_tbl_num_of_reanimation`
/ *Condition*: required / *Type*: int /
Counter of target reanimation during execution.

Returns:

(no returns)

4.1.16 Method: vCreateCCRdata

Create CCR data per test case.

Arguments:

- `_tbl_test_case_id`
/ *Condition*: required / *Type*: int /
test case ID.

- `lCCRdata`
/ *Condition*: required / *Type*: list /
list of CCR data.

Returns:*(no returns)***4.1.17 Method: vFinishTestResult**

Finish upload:

- First do bulk insert of rest of test cases if buffer is not empty.
- Then set state to "new report".

Arguments:

- `tbl_test_result_id`
/ *Condition*: required / *Type*: str /
UUID of test result.

Returns:*(no returns)***4.1.18 Method: vUpdateEvtbls**Call `update_evtbls` stored procedure.**Arguments:***(no arguments)***Returns:***(no returns)***4.1.19 Method: vUpdateEvtbl**Call `update_evtbl` stored procedure to update provided `test_result_id`.**Arguments:**

- `tbl_test_result_id`
/ *Condition*: required / *Type*: str /
UUID of test result.

Returns:*(no returns)***4.1.20 Method: vEnableForeignKeyCheck**Switch `foreign_key_checks` flag.**Arguments:**

- `enable`
/ *Condition*: optional / *Type*: bool / *Default*: True /
If True, enable foreign key constraint.

Returns:*(no returns)*

4.1.21 Method: sGetLatestFileID

Get latest file ID from `tbl_file` table.

Arguments:

- `tbl_test_result_id`
/ *Condition*: required / *Type*: str /
UUID of test result.

Returns:

- `tbl_file_id`
/ *Type*: int /
File ID.

4.1.22 Method: vUpdateFileEndTime

Update test file end time.

Arguments:

- `tbl_file_id`
/ *Condition*: required / *Type*: int /
File ID to be updated.
- `tbl_file_time_end`
/ *Condition*: required / *Type*: str /
File end time as format `%Y-%m-%d %H:%M:%S`.

Returns:

(no returns)

4.1.23 Method: vUpdateResultEndTime

Update test result end time.

Arguments:

- `tbl_test_result_id`
/ *Condition*: required / *Type*: str /
Result UUID to be updated.
- `tbl_result_time_end`
/ *Condition*: required / *Type*: str /
Result end time as format `%Y-%m-%d %H:%M:%S`.

Returns:

(no returns)

4.1.24 Method: bExistingResultID

Verify the given test result UUID is existing in `tbl_result` table or not.

Arguments:

- `tbl_test_result_id`
/ *Condition*: required / *Type*: str /
Result UUID to be verified.

Returns:

- `bExisting`
/ *Type*: bool /
True if test result UUID is already existing.

4.1.25 Method: `arGetProjectVersionSWByID`

Get the project and version_sw information of given `test_result_id`

Arguments:

- `_tbl_test_result_id`
/ *Condition*: required / *Type*: str /
Result UUID to be get the information.

Returns:

- / *Type*: tuple /
None if test result UUID is not existing, else the tuple which contains project and version_sw: (project, variant) is returned.

Chapter 5

RestApiDBAccess.py

5.1 Class: RestApiDBAccess

Imported by:

```
from TestResultDBAccess.DBAccess.RestApiDBAccess import RestApiDBAccess
```

- 5.1.1 Method: `get_certs_file`
- 5.1.2 Method: `encrypt_password`
- 5.1.3 Method: `connect`
- 5.1.4 Method: `disconnect`
- 5.1.5 Method: `arGetCategories`
- 5.1.6 Method: `bExistingResultID`
- 5.1.7 Method: `sGetLatestFileID`
- 5.1.8 Method: `arGetProjectVersionSWByID`
- 5.1.9 Method: `sCreateNewTestResult`
- 5.1.10 Method: `nCreateNewFile`
- 5.1.11 Method: `vCreateNewHeader`
- 5.1.12 Method: `nCreateNewSingleTestCase`
- 5.1.13 Method: `nCreateNewTestCase`
- 5.1.14 Method: `vCreateAbortReason`
- 5.1.15 Method: `vCreateCCRdata`
- 5.1.16 Method: `vCreateTags`
- 5.1.17 Method: `vCreateReanimation`
- 5.1.18 Method: `vSetCategory`
- 5.1.19 Method: `vUpdateFileEndTime`
- 5.1.20 Method: `vUpdateResultEndTime`
- 5.1.21 Method: `vFinishTestResult`
- 5.1.22 Method: `vUpdateEvtbl`
- 5.1.23 Method: `vUpdateEvtbls`

Chapter 6

DBAccessFactory.py

6.1 Class: DBAccessFactory

Imported by:

```
from TestResultDBAccess.DBAccessFactory import DBAccessFactory
```

6.1.1 Method: create

Chapter 7

Appendix

About this package:

Table 7.1: Package setup

Setup parameter	Value
Name	TestResultDBAccess
Version	0.1.0
Date	25.04.2024
Description	Imports robot result(s) to TestResultWebApp database
Package URL	robotframework-robotlog2db
Author	Tran Duy Ngoan
Email	Ngoan.TranDuy@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 8

History

0.1.0	07/2022
<i>Initial version</i>	
1.2.1	22.08.2022
<i>Rework repository's document bases on GenPackageDoc</i>	
1.2.2	13.10.2022
<ul style="list-style-type: none">- Fix findings and enhance README and document files- Change argument name 'outputfile' to 'resultxmlfile'	
1.2.3	10.11.2022
<i>Rename package to TestResultDBAccess</i>	
1.2.4	18.11.2022
<i>Add -append argument which allow to append into existing UUID</i>	
1.3.0	06.12.2022
<ul style="list-style-type: none">- Rework optional arguments in command line- Improve coding: logging messages, shorten variable names, ...	
1.3.1	09.01.2023
<i>Improve messages when verify configuration json file</i>	
1.3.2	28.02.2023
<ul style="list-style-type: none">- Rename key to 'components' in configuration json file- Change implementation of append mode to raise errors without proper given UUID- Enhance console log with component and append mode information- Add more supported keys in configuration json file	
1.3.3	01.03.2023
<i>Add a validation for xmlresultfile with robot schema</i>	
1.3.4	13.03.2023
<ul style="list-style-type: none">- Add a validation for existing project/variant and version_sw in db when using append mode- Remove db maxlength handlers: truncations and validation	
1.3.5	21.03.2023
<i>Support 4 byte characters for importing to db</i>	
1.3.6	05.04.2023
<ul style="list-style-type: none">- Enhance console log with test case counter and component statistics- Add try/except for database access	
1.3.7	14.06.2023
<i>Update README with new instruction for installation and image's links</i>	

1.3.8	21.06.2023
<i>Fix issue with suite.starttime at root suite is N/A</i>	
1.3.9	23.06.2023
<ul style="list-style-type: none">- Retrieve start/end time for suite- Update result schema to support Robotframework Version 6.1	
1.4.0	19.09.2023
<i>Adaption for Robotframework 6.1 with change of <code>TestSuite.source</code> datatype</i>	
1.4.1	15.03.2024
<i>Update <code>robot.xsd</code> schema to support the new log level <code>USER</code> of Robotframework</i>	
1.5.0	22.04.2024
<i>Add implementation of **REST API** interface</i>	