

RobotResults2DB

v. 1.2.1

Tran Duy Ngoan

22.08.2022

Contents

1	Introduction	1
2	Description	2
2.1	Robot Framework Testcase Settings:	2
2.2	Sample Robot Framework Testcase:	3
2.3	Display on WebApp:	4
2.4	Notes:	4
3	CDataBase.py	6
3.1	Class: CDataBase	6
3.1.1	Method: connect	6
3.1.2	Method: disconnect	7
3.1.3	Method: cleanAllTables	7
3.1.4	Method: sCreateNewTestResult	7
3.1.5	Method: nCreateNewFile	8
3.1.6	Method: vCreateNewHeader	9
3.1.7	Method: nCreateNewSingleTestCase	11
3.1.8	Method: nCreateNewTestCase	12
3.1.9	Method: vCreateTags	13
3.1.10	Method: vSetCategory	13
3.1.11	Method: vUpdateStartEndTime	14
3.1.12	Method: arGetCategories	14
3.1.13	Method: vCreateAbortReason	14
3.1.14	Method: vCreateReanimation	15
3.1.15	Method: vCreateCCRdata	15
3.1.16	Method: vFinishTestResult	15
3.1.17	Method: vUpdateEvtbls	15
3.1.18	Method: vUpdateEvtbl	16
3.1.19	Method: vEnableForeignKeyCheck	16
3.1.20	Method: sGetLatestFileID	16
3.1.21	Method: vUpdateFileEndTime	16
3.1.22	Method: vUpdateResultEndTime	17
3.1.23	Method: bExistingResultID	17
4	robot2db.py	18
4.1	Function: is_valid_uuid	18
4.2	Function: get_from_tags	18
4.3	Function: get_branch_from_swversion	19

4.4	Function: <code>format_time</code>	19
4.5	Function: <code>process_suite_metadata</code>	19
4.6	Function: <code>process_metadata</code>	20
4.7	Function: <code>process_suite</code>	20
4.8	Function: <code>process_test</code>	21
4.9	Function: <code>process_config_file</code>	21
4.10	Function: <code>validate_config</code>	22
4.11	Function: <code>normalize_path</code>	22
4.12	Function: <code>truncate_string</code>	23
4.13	Function: <code>RobotResults2DB</code>	23
4.14	Class: <code>Logger</code>	24
4.14.1	Method: <code>config</code>	24
4.14.2	Method: <code>log</code>	24
4.14.3	Method: <code>log_warning</code>	25
4.14.4	Method: <code>log_error</code>	25
5	Appendix	26
6	History	27

Chapter 1

Introduction

[RobotResults2DB](#) tool helps to import robot *output.xml* result file(s) to [TestResultWebApp](#)'s database for presenting an overview about the execution and detail of each test result.

In order to display the Robot Framework results on [TestResultWebApp](#) Dashboard properly, Robot testcase need to give some required information for management such as project/variant, software version, component, ...

Therefore, `Metadata` and `[Tags]` are used to provide that information to *output.xml* result which is used for importing data to WebApp.

Chapter 2

Description

2.1 Robot Framework Testcase Settings:

For the whole test execution:

- Project/Variant (can be overwritten by argument `--variant` or `--config` of [RobotResults2DB](#) tool when importing):

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

- Versions (can be overwritten by argument `--versions` or `--config` of [RobotResults2DB](#) tool when importing):

```
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 (can be overwritten by argument `--config` of [RobotResults2DB](#) tool when importing):

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

- Test Tool - 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}
```

2.2 Sample Robot Framework Testcase:

For test case management, we need some traceable information such as version, testcase ID, component, ... to manage and track testcase(s) on RQM.

So, this information can be provided in `Metadata` (for the whole testsuite/execution info: version, build, ...) and `[Tags]` information (for specific testcase info: component, testcase ID, requirement ID, ...).

Sample Robot Framework testcase with the necessary information for importing to RQM:

```
*** 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 3.2rc2 (Python 3.9.0 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



You don't need to define above highlighted `Metadata` (`testtool` , `machine` and `tester`) with RobotFramework AIO because `RobotFramework.Testsuites` library will handle these definitions within `Suite Setup` .

2.3 Display on WebApp:

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

Above settings in robot testcase will be reflect on **Dashboard** (General view) and **Data table** (Detailed view) as below figures:

Execution result metadata:

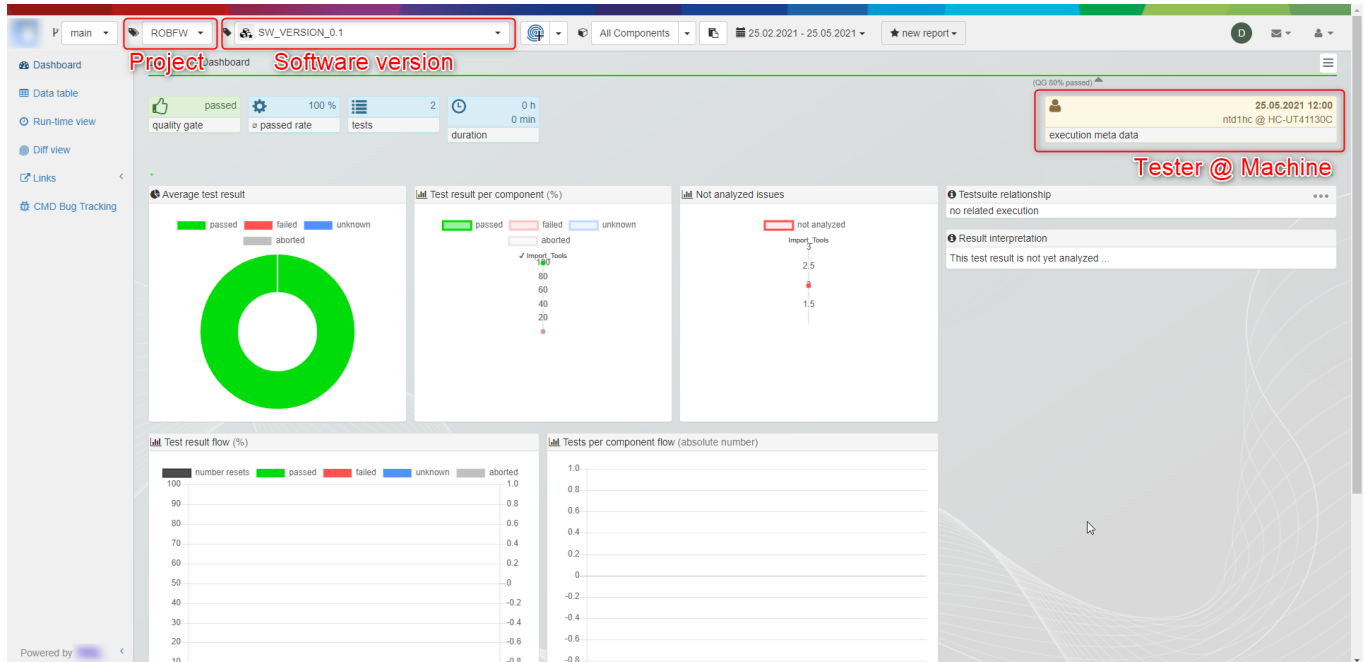


Figure 2.1: Dashboard view

Suite/File metadata and Testcase information:

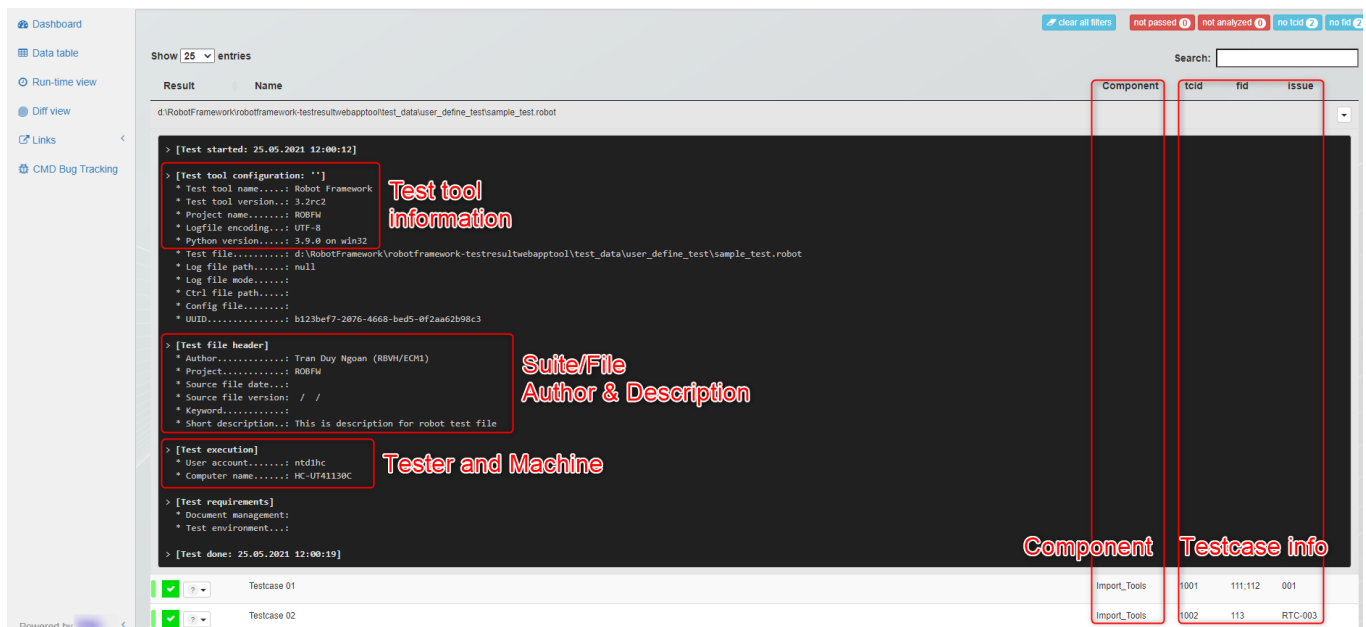


Figure 2.2: Datatable view

2.4 Notes:

When above settings is missing, that leads to the missing information in the *output.xml*.

Some required fields for management will be set to default value when importing with [RobotResults2DB](#) 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.

But, you can provide them as command arguments when executing the [RobotResults2DB](#) tool with below optional arguments (refer its [usage](#)):

- `--variant VARIANT`

To specify the Project/Variant information.

- `--versions VERSIONS`

To specify the Software version information.

- `--config CONFIG`

Provide a configuration json file `CONFIG` which helps:

- To configure the Project/Variant, Software version information (lower priority than above commandline arguments)
- To create a mapping between testcase folder and Component information which is display on [TestResultWebApp](#).

Sample configuration json file:

```
{
  "component" : {
    "cli"      : "robot/cli",
    "core"     : "robot/core",
    "external" : "robot/external",
    "keywords" : "robot/keywords",
    "libdoc"   : "robot/libdoc",
    "output"   : "robot/output",
    "parsing"  : "robot/parsing",
    "reboot"   : "robot/reboot",
    "rpa"      : "robot/rpa",
    "running"  : "robot/running",
    "std_lib"  : "robot/standard_libraries",
    "tags"     : "robot/tags",
    "test_lib" : "robot/test_libraries",
    "testdoc"  : "robot/testdoc",
    "tidy"     : "robot/tidy",
    "variables" : "robot/variables"
  },
  "version_sw" : "Atest",
  "variant"    : "ROBFW"
}
```


Chapter 3

CDataBase.py

3.1 Class: CDataBase

```
RobotResults2DB.CDataBase
```

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

3.1.1 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)

3.1.2 Method: disconnect

Disconnect from TestResultWebApp's database.

Arguments:

(no arguments)

Returns:

(no returns)

3.1.3 Method: cleanAllTables

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

Arguments:

(no arguments)

Returns:

(no returns)

3.1.4 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_swtest`
/ *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.

3.1.5 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.

3.1.6 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)*

3.1.7 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.

3.1.8 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 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:*(no returns)***3.1.9 Method: vCreateTags**

Create tag entries.

Arguments:

- `tbl.test.result.id`
/ *Condition*: required / *Type*: str /
UUID of test result.
- `tbl.user.result.tags`
/ *Condition*: required / *Type*: str /
User tags information.

Returns:*(no returns)***3.1.10 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)*

3.1.11 Method: vUpdateStartTime

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)

3.1.12 Method: arGetCategories

Get existing categories.

Arguments:

(no arguments)

Returns:

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

3.1.13 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)

3.1.14 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)

3.1.15 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)

3.1.16 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)

3.1.17 Method: vUpdateEvtbls

Call `update_evtbls` stored procedure.

Arguments:

(no arguments)

Returns:

(no returns)

3.1.18 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)

3.1.19 Method: vEnableForeignKeyCheck

Switch `foreign_key_checks` flag.

Arguments:

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

Returns:

(no returns)

3.1.20 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.

3.1.21 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)

3.1.22 Method: vUpdateResultEndTime

Update test result end time.

Arguments:

- `_tbl_test_result_id`
/ *Condition*: required / *Type*: int /
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)

3.1.23 Method: bExistingResultID

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

Arguments:

- `_tbl_test_result_id`
/ *Condition*: required / *Type*: int /
Result UUID to be verified.

Returns:

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

Chapter 4

robot2db.py

4.1 Function: is_valid_uuid

Verify the given UUID is valid or not.

Arguments:

- `uuid_to_test`
/ *Condition*: required / *Type*: str /
UUID to be verified.
- `version`
/ *Condition*: optional / *Type*: int / *Default*: 4 /
UUID version.

Returns:

- `bValid`
/ *Type*: bool /
True if the given UUID is valid.

4.2 Function: get_from_tags

Extract testcase information from tags.

Example: TCID-xxxx, FID-xxxx, ...

Arguments:

- `lTags`
/ *Condition*: required / *Type*: list /
List of tag information.
- `reInfo`
/ *Condition*: required / *Type*: str /
Regex to get the expected info (ID) from tag info.

Returns:

- `lInfo`
/ *Type*: list /
List of expected information (ID)

4.3 Function: get_branch_from_swversion

Get branch name from software version information.

Convention of branch information in suffix of software version:

- All software version with .0F is the main/freature branch. The leading number is the current year. E.g. 17.0F03
- All software version with .1S, .2S, ... is a stabi branch. The leading number is the year of branching out for stabilization. The number before "S" is the order of branching out in the year.

Arguments:

- `sw_version`
/ *Condition*: required / *Type*: str /
Software version.

Returns:

- `branch_name`
/ *Type*: str /
Branch name.

4.4 Function: format_time

Format the given time string to TestResultWebApp's format for importing to db.

Arguments:

- `stime`
/ *Condition*: required / *Type*: str /
String of time.

Returns:

- `sFormattedTime`
/ *Type*: str /
TestResultWebApp's time as format %Y-%m-%d %H:%M:%S.

4.5 Function: process_suite_metadata

Try to find metadata information from all suite levels.

Metadata at top suite level has a highest priority.

Arguments:

- `suite`
/ *Condition*: required / *Type*: TestSuite object /
Robot suite object.
- `default_metadata`
/ *Condition*: optional / *Type*: dict / *Default*: DEFAULT_METADATA /
Initial Metadata information for updating.

Returns:

- `dMetadata`
/ *Type*: dict /
Dictionary of Metadata information.

4.6 Function: process_metadata

Extract metadata from suite result bases on DEFAULT_METADATA.

Arguments:

- `metadata`
/ *Condition*: required / *Type*: dict /
Robot metadata object.
- `default_metadata`
/ *Condition*: optional / *Type*: dict / *Default*: DEFAULT_METADATA /
Initial Metadata information for updating.

Returns:

- `dMetadata`
/ *Type*: dict /
Dictionary of Metadata information.

4.7 Function: process_suite

Process to the lowest suite level (test file):

- Create new file and its header information
- Then, process all child test cases

Arguments:

- `db`
/ *Condition*: required / *Type*: CDataBase object /
CDataBase object.
- `suite`
/ *Condition*: required / *Type*: TestSuite object /
Robot suite object.
- `tbltest_result_id`
/ *Condition*: required / *Type*: str /
UUID of test result for importing.
- `root_metadata`
/ *Condition*: required / *Type*: dict /
Metadata information from root level.
- `dConfig`
/ *Condition*: required / *Type*: dict / *Default*: None /
Configuration data which is parsed from given json configuration file.

Returns:

(no returns)

4.8 Function: process_test

Process test case data and create new test case record.

Arguments:

- db
/ Condition: required / Type: CDataBase object /
CDataBase object.
- test
/ Condition: required / Type: TestCase object /
Robot test object.
- file_id
/ Condition: required / Type: int /
File ID for mapping.
- test_result_id
/ Condition: required / Type: str /
Test result ID for mapping.
- metadata_info
/ Condition: required / Type: dict /
Metadata information.
- test_number
/ Condition: required / Type: int /
Order of test case in file.

Returns:

(no returns)

4.9 Function: process_config_file

Parse information from configuration file:

- component:

```
{
  "component" : {
    "componentA" : "componentA/path/to/testcase",
    "componentB" : "componentB/path/to/testcase",
    "componentC" : [
      "componentC1/path/to/testcase",
      "componentC2/path/to/testcase"
    ]
  }
}
```

Then all testcases which their paths contain componentA/path/to/testcase will be belong to componentA, ...

- variant, version-sw: configuration file has low priority than command line.

Arguments:

- `config.file`
/ *Condition*: required / *Type*: str /
Path to configuration file.

Returns:

- `dConfig`
/ *Type*: dict /
Configuration object.

4.10 Function: `validate_config`

Validate the json configuration base on given schema.

Default schema just supports component, variant and version_sw.

```
CONFIG_SCHEMA = {
    "component" : [str, dict],
    "variant"   : str,
    "version_sw": str,
}
```

Arguments:

- `dConfig`
/ *Condition*: required / *Type*: dict /
Json configuration object to be verified.
- `dSchema`
/ *Condition*: optional / *Type*: dict / *Default*: CONFIG_SCHEMA /
Schema for the validation.
- `bExitOnFail`
/ *Condition*: optional / *Type*: bool / *Default*: True /
If True, exit tool in case the validation is fail.

Returns:

- `bValid`
/ *Type*: bool /
True if the given json configuration data is valid.

4.11 Function: `normalize_path`

Normalize path file.

Arguments:

- `sPath`
/ *Condition*: required / *Type*: str /
Path file to be normalized.
- `sNPath`
/ *Type*: str /
Normalized path file.

4.12 Function: truncate_string

Truncate input string before importing to database.

Arguments:

- sString
/ *Condition*: required / *Type*: str /
Input string for truncation.
- iMaxLength
/ *Condition*: required / *Type*: int /
Max length of string to be allowed.
- sEndChars
/ *Condition*: optional / *Type*: str / *Default*: '...' /
End characters which are added to end of truncated string.

Returns:

- content
/ *Type*: str /
String after truncation.

4.13 Function: RobotResults2DB

Import robot results from `output.xml` to `TestResultWebApp`'s database.

Flow to import Robot results to database:

1. Process provided arguments from command line.
2. Connect to database.
3. Parse Robot results.
4. Import results into database.
5. Disconnect from database.

Arguments:

- args
/ *Condition*: required / *Type*: ArgumentParser object /
Argument parser object which contains:
 - outputfile : path to the output file or directory with output 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 name.
 - recursive : if True, then the path is searched recursively for log files to be imported.
 - dryrun : if True, then just check the RQM authentication and show what would be done.
 - UUID : UUID used to identify the import and version ID on `TestResultWebApp`.
 - variant : variant name to be set for this import.
 - versions : metadata: Versions (Software;Hardware;Test) to be set for this import.
 - config : configuration json file for component mapping information.

Returns:

(no returns)

4.14 Class: Logger

```
RobotResults2DB.robot2db
```

Logger class for logging message.

4.14.1 Method: config

Configure Logger class.

Arguments:

- `output_console`
/ *Condition*: optional / *Type*: bool / *Default*: True /
Write message to console output.
- `output_logfile`
/ *Condition*: optional / *Type*: str / *Default*: None /
Path to log file output.
- `indent`
/ *Condition*: optional / *Type*: int / *Default*: 0 /
Offset indent.
- `dryrun`
/ *Condition*: optional / *Type*: bool / *Default*: True /
If set, a prefix as 'dryrun' is added for all messages.

Returns:

(no returns)

4.14.2 Method: log

Write log message to console/file output.

Arguments:

- `msg`
/ *Condition*: optional / *Type*: str / *Default*: " /
Message which is written to output.
- `color`
/ *Condition*: optional / *Type*: str / *Default*: None /
Color style for the message.
- `indent`
/ *Condition*: optional / *Type*: int / *Default*: 0 /
Offset indent.

Returns:

(no returns)

4.14.3 Method: `log_warning`

Write warning message to console/file output.

Arguments:

- `msg`
/ *Condition*: required / *Type*: str /
Warning message which is written to output.

Returns:

(no returns)

4.14.4 Method: `log_error`

Write error message to console/file output.

Arguments:

- `msg`
/ *Condition*: required / *Type*: str /
Error message which is written to output.
- `fatal_error`
/ *Condition*: optional / *Type*: bool / *Default*: False /
If set, tool will terminate after logging error message.

Returns:

(no returns)

Chapter 5

Appendix

About this package:

Table 5.1: Package setup

Setup parameter	Value
Name	RobotResults2DB
Version	1.2.1
Date	22.08.2022
Description	Imports robot result(s) to TestResultWebApp database
Package URL	robotframework-testresultwebapptool
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 6

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>	

RobotResults2DB.pdf*Created at 25.08.2022 - 15:55:41**by GenPackageDoc v. 0.27.0*
