RobotLog2RQM

v. 1.1.4

Tran Duy Ngoan

10.11.2022

Contents

Chapter 1

Introduction

RobotLog2RQM tool provides the ability to interact with RQM resources (test plan, test case, build, ...).

RobotLog2RQM tool uses RqmAPI to:

- get resource: by given ID or all vailable entities of resource type.
- update resource: by given ID.
- create new resource: with resource templates under RQM_templates folder

Chapter 2

Description

2.1 Sample Robot Framework Testcase:

For test case management, we need some tracable 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 neccessary information for importing to RQM:

```
*** Settings ***
Metadata    project       ROBFW              # Test Environment
Metadata    version_sw    SW_VERSION_0.1     # Build Record
Metadata    machine       %{COMPUTERNAME}    # Hostname
Metadata    component     Import_Tools       # Component - is used for test case
Metadata    team-area     Internet Team RQM  # team-area (case-sensitive)

*** Test Cases ***
Testcase 01
   [Documentation]   This test is traceable with provided tcid
   [Tags]   TCID-1001   FID-112   FID-111    ↩
    ↪ robotfile-https://github.com/test-fullautomation
   Log      This is Testcase 01

Testcase 02
   [Documentation]  This new testcase will be created if -createmissing argument
                ...  is provided when importing
   [Tags]   FID-113  robotfile-https://github.com/test-fullautomation
   Log      This is Testcase 02
```

Listing 2.1: Sample Robot Framework testcase

> **Hint**
>
> In case you are using RobotFramework AIO with RobotFramework_Testsuites library for executing Robot testcase(s). You do not need to define above highlighted `Metadata` in your Robot test case.
> These `Metadata` information will be defined implicitly within the `Suite Setup` bases on your environment and configuration in *.json file.

2.2 Tool features:

By default, tool will base on provided arguments (see Usage) and `tcid` information in Robot test case(s) to:

- Login the RQM server and verify the provided `project` , `testplan` .
- Create build record and test environment (if already provided in Robot test case and not existing on RQM) for execution.

- Create new Test Case Execution Record - TCER (if it is not existing) bases on test case ID and `testplan` ID.
- Create new Test Case Execution Result which contents the detail and result state of test case.
- Link all test case(s) to provided `testplan` .

Besides, robotframework-robotlog2rqm tool also supports to:

- Create new test case(s) which is not existing on RQM (do not have `tcid` information) while importing result(s) to RQM with optional argument `-createmissing` .
- Update information (such as testcase name, Requirement ID, ...) of existing testcase on RQM with optional argument `-updatetestcase` .

2.3 Robot Testcase information on RQM:

For more detail how the Robot Framework testcase information is displayed on RQM, please refer be mapping table:

RQM data		Robot Framework	
Resource	**Attribute/ Field**	**Testsuite/Testcase**	**Output.xml**
Build Record	Title	`Metadata version_sw Build`	**//suite/metadata/item[@name="version_sw"]**
Test Environment	Title	`Metadata project Environment`	**//suite/metadata/item[@name="project"]**
Test Case	ID	`[Tags] tcid-xxx`	**//suite/test/tags/tag[@text="tcid-xxx"]**
	Name	tesname	**//suite/test/@name**
	Team Area	`Metadata team-area Team_Area`	**//suite/metadata/item[@name="team-area"]**
	Description	test doc - `[Documentation]`	**//suite/test/doc/@text**
	Owner	provided `user` in cli	
	Component/ Categories	`Metadata component Component`	**//suite/metadata/item[@name="component"]**
	Requirement ID	`[Tags] fid-yyy`	**//suite/test/tags/tag[@text="fid-yyy"]**
	Robot File	`[Tags] robotfile-zzz`	**//suite/test/tags/tag[@text="robotfile-zzz"]**
Test Case Execution Record (TCER)	Owner	provided `user` in cli	
	Team Area	`Metadata team-area Team_Area`	**//suite/metadata/item[@name="team-area"]**
	Test Plan	Interaction URL to provided `testplan` in cli	
	Test Case	Interaction URL to provided testcase ID: provided tcid in `[Tags]: tcid-xxx` or generated tcid when using `-createmissing`	**//suite/test/tags/tag[@text="tcid-xxx"]**
	Test Environment	`Metadata project Environment`	**//suite/metadata/item[@name="project"]**
Test Result	Owner	provided `user` in cli	
	Tested By	provided `user` in cli - userid must be used	
	Team Area	`Metadata team-area Team_Area`	**//suite/metadata/item[@name="team-area"]**
	Actual Result	Test case result (PASSED, FAILED, UNKNOWN)	**//suite/test/status/@status**
	Host Name	`Metadata machine ↩ ↪ %{COMPUTERNAME}`	**//suite/metadata/item[@name="machine"]**
	Test Plan	Interaction URL to provided `testplan` in cli	
	Test Case	Interaction URL to provided testcase ID: provided tcid in `[Tags]: tcid-xxx` or generated tcid when using `-createmissing`	**//suite/test/tags/tag[@text="tcid-xxx"]**
	Test Case Execution Record	Interaction URL to TCER ID	
	Build	`Metadata version_sw Build`	**//suite/metadata/item[@name="version_sw"]**
	Start Time	Test case start time	**//suite/test/status/@starttime**
	End Time	Test case end time	**//suite/test/status/@endtime**
	Total Run Time	Calculated from start and end time	
	Result Details	Test case message log	**//suite/test/status/@text**

Table 2.1: RQM data & Robot Framework testcase/output.xml

Chapter 3

CRQM.py

3.1 Function: get_xml_tree

Parse xml object from file.

Arguments:

- `file_name`
  / *Condition*: required / *Type*: str /
  Path to file or file-like object.
- `bdtd_validation`
  / *Condition*: optional / *Type*: bool /
  If True, validate against a DTD referenced by the document.

Returns:

- `oTree`
  / *Type*: lxml.etree._ElementTree object /
  The xml etree object.

3.2 Class: CRQMClient

Imported by:

```
from RobotLog2RQM.CRQM import CRQMClient
```

CRQMClient class uses RQM REST APIs to get, create and update resources (testplan, testcase, test result, ...) on RQM - Rational Quality Manager

Resoure type mapping:

- buildrecord: Build Record
- configuration: Test Environment
- testplan: Test Plan
- testsuite: Test Suite
- suiteexecutionrecord: Test Suite Execution Record (TSER)
- testsuitelog: Test Suite Log
- testcase: Test Case
- executionworkitem: Test Execution Record (TCER)
- executionresult: Execution Result

3.2.1 Method: login

Log in RQM by provided user & password.

Arguments:

(*no arguments*)

Returns:

- `bSuccess`
  / *Type*: bool /
  Indicates if the computation of the method `login` was successful or not.

3.2.2 Method: verifyProjectName

Verify the project name by searching it in project-areas XML response.

Arguments:

(*no arguments*)

Returns:

- `bSuccess`
  / *Type*: bool /
  Indicates if the computation of the method `verifyProjectName` was successful or not.

3.2.3 Method: disconnect

Disconnect from RQM.

Arguments:

(*no arguments*)

Returns:

(*no returns*)

3.2.4 Method: config

Configure RQMClient with testplan ID, build, configuration, createmissing, ...

- Verify the existence of provided testplan ID.
- Verify the existences of provided build and configuration names before creating new ones.

Arguments:

- `plan_id`
  / *Condition*: required / *Type*: str /
  Testplan ID of RQM project for importing result(s).
- `build_name`
  / *Condition*: optional / *Type*: str / *Default*: None /
  The Build Record for linking result(s). Set it to None if not be used, the empty name '' will lead to error.
- `config_name`
  / *Condition*: optional / *Type*: str / *Default*: None /
  The Test Environment for linking result(s). Set it to None if not be used, the empty name '' may lead to error.
- `createmissing`
  / *Condition*: optional / *Type*: bool / *Default*: False /
  If True, the testcase without tcid information will be created on RQM.

- `updatetestcase`
  / *Condition*: optional / *Type*: bool / *Default*: False /
  If True, the information of testcase on RQM will be updated bases on robot testfile.
- `suite_id (optional)`
  / *Condition*: optional / *Type*: str / *Default*: None /
  Testsuite ID of RQM project for importing result(s).

Returns:
(*no returns*)

3.2.5 Method: userURL

Return interaction URL of provided userID

Arguments:

- `userID`
  / *Condition*: required / *Type*: str /
  The user ID.

Returns:

- `userURL`
  / *Type*: str /
  The interaction URL of provided userID.

3.2.6 Method: integrationURL

Return interaction URL of provided reource and ID. The provided ID can be internalID (contains only digits) or externalID.

Arguments:

- `resourceType`
  / *Condition*: required / *Type*: str /
  The RQM resource type (e.g: "testplan", "testcase", ...).
- `id`
  / *Condition*: optional / *Type*: str / *Default*: None /
  The ID of given resource.
  - If given: the specified url to resource ID is returned.
  - If `None`: the url to resource type (to get all entity) is returned.
- `forceinternalID`
  / *Condition*: optional / *Type*: bool / *Default*: False /
  If True, force to return the url of resource as internal ID.

Returns:

- `integrationURL`
  / *Type*: str /
  The interaction URL of provided reource and ID.

3.2.7 Method: webIDfromResponse

Get internal ID (number) from response of POST method.

Note: Only executionresult has response text. Other resources has only response header.

Arguments:

- `response`
  / *Condition*: required / *Type*: str /
  The xml response from POST method for parsing ID information.
- `tagID`
  / *Condition*: optional / *Type*: str / *Default*: 'rqm:resultId' /
  Tag name which contains ID information.

Returns:

- `resultId`
  / *Type*: str /
  The internal ID (as number).

3.2.8 Method: webIDfromGeneratedID

Return web ID (ns2:webId) from generate ID by get resource data from RQM.
Note:

- This method is only used for generated testcase, executionworkitem and executionresult.
- buildrecord and configuration does not have ns2:webId in response data.

Arguments:

- `resourrceType`
  / *Condition*: required / *Type*: str /
  The RQM resource type.
- `generateID`
  / *Condition*: required / *Type*: str /
  The Slug ID which is returned in Content-Location from POST response.

Returns:

- `webID`
  / *Type*: str /
  The web ID (as number).

3.2.9 Method: getResourceByID

Return data of provided resource and ID by GET method
Arguments:

- `resourrceType`
  / *Condition*: required / *Type*: str /
  The RQM resource type.

- `id`
  / *Condition*: required / *Type*: str /
  ID of resource.

Returns:

- `res`
  / *Type*: Response object /
  Response data of GET request.

3.2.10 Method: getAllByResource

Return all entries (in all pages) of provided resource by GET method.

Arguments:

- `resourrceType`
  / *Condition*: required / *Type*: str /
  The RQM resource type.

Returns:

- `dReturn`
  / *Type*: dict /
  A dictionary which contains response status, message and data.
  Example:

```
{
   'success' : False,
   'message' : '',
   'data'    : {}
}
```

3.2.11 Method: getAllBuildRecords

Get all available build records of project on RQM and store them into dBuildVersion property.

Arguments:

(*no arguments*)

Returns:

(*no returns*)

3.2.12 Method: getAllConfigurations

Get all available configurations of project on RQM and store them into dConfiguation property.

Arguments:

(*no arguments*)

Returns:

(*no returns*)

3.2.13 Method: getAllTeamAreas

Get all available team-areas of project on RQM and store them into dTeamAreas property.

Example:

```
{
   'teamA' : '{host}/qm/process/project-areas/{project-id}/team-areas/{teamA-id}',
   'teamB' : '{host}/qm/process/project-areas/{project-id}/team-areas/{teamB-id}'
}
```

Arguments:

(*no arguments*)

Returns:

(*no returns*)

3.2.14 Method: addTeamAreaNode

Append team-area node which contains URL to given team-area into xml template.

Note: team-area information is case-casesensitive

Arguments:

- `root`
  / *Condition*: required / *Type*: Element object /
  The xml root object.
- `sTeam`
  / *Condition*: required / *Type*: str /
  Team name to be added.

Returns:

- `root`
  / *Type*: str /
  The xml root object with addition team-area node.

3.2.15 Method: createTestcaseTemplate

Return testcase template from provided information.

Arguments:

- `testcaseName`
  / *Condition*: required / *Type*: str /
  Testcase name.
- `sDescription`
  / *Condition*: optional / *Type*: str / *Default*: '' /
  Testcase description.
- `sComponent`
  / *Condition*: optional / *Type*: str / *Default*: '' /
  Component which testcase is belong to.
- `sFID`
  / *Condition*: optional / *Type*: str / *Default*: '' /
  Function ID (requirement ID) for linking.
- `sTeam`
  / *Condition*: optional / *Type*: str / *Default*: '' /
  Team name for linking.

- `sRobotFile`
  / *Condition*: optional / *Type*: str / *Default*: '' /
  Link to robot file on source control.
- `sTestType`
  / *Condition*: optional / *Type*: str / *Default*: '' /
  Test type information.
- `sASIL`
  / *Condition*: optional / *Type*: str / *Default*: '' /
  ASIL information.
- `sOwnerID`
  / *Condition*: optional / *Type*: str / *Default*: '' /
  User ID of testcase owner.
- `sTCtemplate`
  / *Condition*: optional / *Type*: str / *Default*: None /
  Existing testcase template as xml string.
  If not provided, template file under RQM_templates is used as default.

Returns:

- `sTCxml`
  / *Type*: str /
  The xml testcase template as string.

3.2.16 Method: createTCERTemplate

Return testcase execution record template from provided information.

Arguments:

- `testcaseID`
  / *Condition*: required / *Type*: str /
  Testcase ID for linking.
- `testcaseName`
  / *Condition*: required / *Type*: str /
  Testcase name.
- `testplanID`
  / *Condition*: required / *Type*: str /
  Testplan ID for linking.
- `confID`
  / *Condition*: optional / *Type*: str / *Default*: '' /
  Configuration - Test Environment for linking.
- `sTeam`
  / *Condition*: optional / *Type*: str / *Default*: '' /
  Team name for linking.
- `sOwnerID`
  / *Condition*: optional / *Type*: str / *Default*: '' /
  User ID of testcase owner.

Returns:

- `sTCERxml`
  / *Type*: str /
  The xml testcase execution record template as string.

3.2.17 Method: createExecutionResultTemplate

Return testcase execution result template from provided information.

Arguments:

- `testcaseID`
  / *Condition*: required / *Type*: str /
  Testcase ID for linking.
- `testcaseName`
  / *Condition*: required / *Type*: str /
  Testcase name.
- `testplanID`
  / *Condition*: required / *Type*: str /
  Testplan ID for linking.
- `TCERID`
  / *Condition*: required / *Type*: str /
  Testcase execution record (TCER) ID for linking.
- `resultState`
  / *Condition*: required / *Type*: str /
  Testcase result status.
- `startTime`
  / *Condition*: required / *Type*: str /
  Testcase start time.
- `endTime`
  / *Condition*: optional / *Type*: str / *Default*: '' /
  Testcase end time.
- `duration`
  / *Condition*: optional / *Type*: str / *Default*: '' /
  Testcase duration.
- `testPC`
  / *Condition*: optional / *Type*: str / *Default*: '' /
  Test PC which executed testcase.
- `testBy`
  / *Condition*: optional / *Type*: str / *Default*: '' /
  User ID who executed testcase.
- `lastlog`
  / *Condition*: optional / *Type*: str / *Default*: '' /
  Traceback information (for Failed testcase).

- `buildrecordID`
  / *Condition*: optional / *Type*: str / *Default*: '' /
  Build Record ID for linking.
- `sTeam`
  / *Condition*: optional / *Type*: str / *Default*: '' /
  Team name for linking.
- `sOwnerID`
  / *Condition*: optional / *Type*: str / *Default*: '' /
  User ID of testcase owner.

Returns:

- `sTCResultxml`
  / *Type*: str /
  The xml testcase result template as string.

3.2.18 Method: createBuildRecordTemplate

Return build record template from provided build name.

Arguments:

- `buildName`
  / *Condition*: required / *Type*: str /
  Build Record name.

Returns:

- `sBuildxml`
  / *Type*: str /
  The xml build template as string.

3.2.19 Method: createConfigurationTemplate

Return configuration - Test Environment template from provided configuration name.

Arguments:

- `buildName`
  / *Condition*: required / *Type*: str /
  Configuration - Test Environment name.

Returns:

- `sEnvironmentxml`
  / *Type*: str /
  The xml test environment template as string.

3.2.20 Method: createTSERTemplate

Return testsuite execution record (TSER) template from provided configuration name.

Arguments:

- `testsuiteID`
  / *Condition*: required / *Type*: str /
  Testsuite ID.
- `testsuiteName`
  / *Condition*: required / *Type*: str /
  Testsuite name.
- `testplanID`
  / *Condition*: required / *Type*: str /
  Testplan ID for linking.
- `confID`
  / *Condition*: optional / *Type*: str / *Default*: '' /
  Configuration - Test Environment ID for linking.
- `sOwnerID`
  / *Condition*: optional / *Type*: str / *Default*: '' /
  User ID of testsuite owner.

Returns:

- `sTSxml`
  / *Type*: str /
  The xml testsuite template as string.

3.2.21 Method: createTestsuiteResultTemplate

Return testsuite execution result template from provided configuration name.

Arguments:

- `testsuiteID`
  / *Condition*: required / *Type*: str /
  Testsuite ID.
- `testsuiteName`
  / *Condition*: required / *Type*: str /
  Testsuite name.
- `TSERID`
  / *Condition*: required / *Type*: str /
  Testsuite execution record (TSER) ID for linking.
- `lTCER`
  / *Condition*: required / *Type*: str /
  List of testcase execution records (TCER) for linking.
- `lTCResults`
  / *Condition*: required / *Type*: str /
  List of testcase results for linking.

- `startTime`
  / *Condition*: optional / *Type*: str / *Default*: '' /
  Testsuite start time.
- `endTime`
  / *Condition*: optional / *Type*: str / *Default*: '' /
  Testsuite end time.
- `duration`
  / *Condition*: optional / *Type*: str / *Default*: '' /
  Testsuite duration.
- `sOwnerID`
  / *Condition*: optional / *Type*: str / *Default*: '' /
  User ID of testsuite owner.

Returns:

- `sTSResultxml`
  / *Type*: str /
  The xml testsuite result template as string.

3.2.22 Method: createResource

Create new resource with provided data from template by POST method.

Arguments:

- `resourceType`
  / *Condition*: required / *Type*: str /
  Resource type.
- `content`
  / *Condition*: required / *Type*: str /
  The xml template as string.

Returns:

- `returnObj`
  / *Type*: dict /
  A dictionary reponse which contains status, ID, status_code and error message.
  Example:

```
{
   'success' : False,
   'id': None,
   'message': '',
   'status_code': ''
}
```

3.2.23 Method: createBuildRecord

Create new build record.

Arguments:

- `sBuildSWVersion`
  / *Condition*: required / *Type*: str /
  Build version - Build Record name.
- `forceCreate`
  / *Condition*: optional / *Type*: bool / *Default*: False /
  If True, force to create new build record without existing verification.

Returns:

- `returnObj`
  / *Type*: dict /
  A dictionary reponse which contains status, ID, status_code and error message.
  Example:

```
{
   'success' : False,
   'id': None,
   'message': '',
   'status_code': ''
}
```

3.2.24 Method: createConfiguration

Create new configuration - test environment.

Arguments:

- `sConfigurationName`
  / *Condition*: required / *Type*: str /
  Configuration - Test Environment name.
- `forceCreate`
  / *Condition*: optional / *Type*: str / *Default*: False /
  If True, force to create new Test Environment without existing verification.

Returns:

- `returnObj`
  / *Type*: dict /
  A dictionary reponse which contains status, ID, status_code and error message.
  Example:

```
{
   'success' : False,
   'id': None,
   'message': '',
   'status_code': ''
}
```

3.2.25 Method: updateResourceByID

Update data of provided resource and ID by PUT method.

Arguments:

- `resourceType`
  / *Condition*: required / *Type*: str /
  Resource type.
- `id`
  / *Condition*: required / *Type*: str /
  Resource id.
- `content`
  / *Condition*: required / *Type*: str /
  The xml template as string.

Returns:

- `res`
  / *Type*: Response object /
  Response object from PUT request.

3.2.26 Method: linkListTestcase2Testplan

Link list of test cases to provided testplan ID.

Arguments:

- `testplanID`
  / *Condition*: required / *Type*: str /
  Testplan ID to link given testcase(s).
- `lTestcases`
  / *Condition*: optional / *Type*: list / *Default*: None /
  List of testcase(s) to be linked with given testplan.
  If not provide, lTestcaseIDs property will be used as list of testcase.

Returns:

- `returnObj`
  / *Type*: dict /
  Response dictionary which contains status and error message.
  Example:

```
{
   'success' : False,
   'message': ''
}
```

3.2.27 Method: linkListTestcase2Testsuite

Link list of test cases to provided testsuite ID

Arguments:

- `testsuiteID`
  / *Condition*: required / *Type*: str /
  Testsuite ID to link given testcase(s).
- `lTestcases`
  / *Condition*: optional / *Type*: list / *Default*: None /
  List of testcase(s) to be linked with given testplan.
  If not provide, lTestcaseIDs property will be used as list of testcase.

Returns:

- `returnObj`
  / *Type*: dict /
  Response dictionary which contains status and error message.
  Example:

```
{
   'success' : False,
   'message': ''
}
```

Chapter 4

robotlog2rqm.py

4.1 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 expectated info (ID) from tag info.

Returns:

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

4.2 Function: convert_to_datetime

Convert time string to datetime.

Arguments:

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

Returns:

- `dt`
  / *Type*: datetime object/
  Datetime object.

4.3 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.

4.4 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.

4.5 Function: process_suite

Process robot suite for importing to RQM.

Arguments:

- `RQMClient`
  / *Condition*: required / *Type*: RQMClient object/
  RQMClient object.
- `suite`
  / *Condition*: required / *Type*: TestSuite object/
  Robot suite object.

Returns:

(*no returns*)

4.6 Function: process_test

Process robot test for importing to RQM.

Arguments:

- `RQMClient`
  / *Condition*: required / *Type*: RQMClient object/
  RQMClient object.
- `test`
  / *Condition*: required / *Type*: TestCase object/
  Robot test object.

Returns:

(*no returns*)

4.7 Function: RobotLog2RQM

Import robot results from output.xml to RQM - IBM Rational Quality Manager.
Flow to import Robot results to RQM:

1. Process provided arguments from command line
2. Login Rational Quality Management (RQM)
3. Parse Robot results
4. Import results into RQM
5. Link all executed testcases to provided testplan/testsuite ID

Arguments:

- `args`
  / *Condition*: required / *Type*: ArgumentParser object /
  Argument parser object which contains:
  - resultxmlfile : path to the xml result file or directory of result files to be imported.
  - host : RQM host url.
  - project : RQM project name.
  - user : user for RQM login.
  - password : user password for RQM login.
  - testplan : RQM testplan ID.
  - recursive : if True, then the path is searched recursively for log files to be imported.
  - createmissing : if True, then all testcases without fcid are created when importing.
  - updatetestcase : if True, then testcases information on RQM will be updated bases on robot testfile.
  - dryrun : if True, then just check the RQM authentication and show what would be done.

Returns:

(*no returns*)

4.8 Class: Logger

Imported by:

```
from RobotLog2RQM.robotlog2rqm import Logger
```

Logger class for logging message.

4.8.1 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*)

4.8.2 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*)

4.8.3 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*)

4.8.4 Method: log_error

Write error message to console/file output.

- `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	RobotLog2RQM
Version	1.1.4
Date	10.11.2022
Description	Imports robot result(s) to IBM Rational Quality Manager (RQM)
Package URL	robotframework-robotlog2rqm
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
Initial version	
1.1.1	01.08.2022
Rework repository's document bases on GenPackageDoc	
1.1.2	25.08.2022
- Correct indent of sourcode's docstring. *- Update new style for history.*	
1.1.3	13.10.2022
- Fix findings and enhance README and document files *- Change argument name 'outputfile' to 'resultxmlfile'*	
1.1.4	10.11.2022
Rename package to RobotLog2RQM	

RobotLog2RQM.pdf

Created at 10.11.2022 - 13:47:50

by GenPackageDoc v. 0.34.0