Squash TF Java Junit Runner Documentation

squashtest

Apr 03, 2020

Contents

CHAPTER 1

Runner Functions

1.1 List implemented Robot Framework tests

This Mojo enables one to list as a Json file the available implemented tests. In order to do so one only needs to run the following command :

```
mvn org.squashtest.ta.galaxia:squash-tf-robotframework-runner-maven-plugin:1.0.0-
↪RELEASE:list
```

The command is structured as follows

- `mvn` : launch Maven
- `org.squashtest.ta.galaxia:squash-tf-robotframework-runner-maven-plugin:1.0.0-RELEASE:list` : the actual listing Mojo provided by the **Squash TF** Robot Framework Runner. It lists all Robot Framework test present in the project.

Note: This listing doesn't list only ".robot" file but also tests cases inside the .robot file.

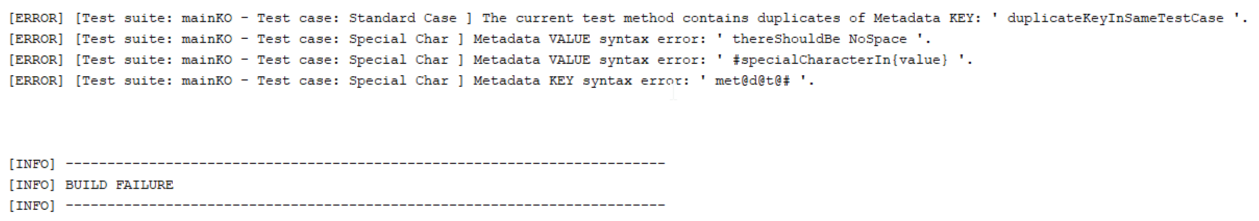
1.1.1 'list' goal with Metadata

If there are Metadata in the current test project, the goal *'list'* searches and checks if all metadata respect the conventions for writing and using Squash TF metadata (See *Metadata in Robot Framework Runner* for more information about Metadata syntax conventions).

The goal will check through the project and collect all the metadata error(s).

If there are any error, the build will fail and a list of the incorrect metadata will be displayed. Otherwise, a **SUCCESS** result will be obtained.

The test suite and test case names are also displayed with the incorrect metadata.



```
[ERROR] [Test suite: mainKO - Test case: Standard Case ] The current test method contains duplicates of Metadata KEY: ' duplicateKeyInSameTestCase '.
[ERROR] [Test suite: mainKO - Test case: Special Char ] Metadata VALUE syntax error: ' thereShouldBe NoSpace '.
[ERROR] [Test suite: mainKO - Test case: Special Char ] Metadata VALUE syntax error: ' #specialCharacterIn{value} '.
[ERROR] [Test suite: mainKO - Test case: Special Char ] Metadata KEY syntax error: ' met@d@t@# '.


[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
```

1.1.2 Listing test JSON report with Metadata

If the build is successful, the generated report (JSON file) will contain the metadata associated with each of the test scripts.

```
{
    "timestamp": "2019-12-11T16:45:40.259+0000",
    "name": "tests",
    "contents": [
        {
            "name": "Main",
            "contents": [
                {
                    "name": "Standard Case",
                    "metadata": {
                        "TC_UID": [
                            "598621846"
                        ],
                        "linked-TC": [
                            "444555552",
                            "465454603",
                            "481289439"
                        ],
                        "metadataWithOneValue": [
                            "12345"
                        ]
                    },
                    "contents": null
                },
                {
                    "name": "Special Char",
                    "metadata": {},
                    "contents": null
                },
                {
                    "name": "No Last Name",
                    "metadata": {},
                    "contents": null
                },
                {
                    "name": "No First Name",
                    "metadata": {},
                    "contents": null
                }
            ]
```

(continues on next page)

(continued from previous page)

```
        }
    ]
}
```

Note: To ignore thoroughly Metadata during the listing process as well as in the report, insert *tf.disableMetadata* property after the goal "list".

```
mvn org.squashtest.ta.galaxia:squash-tf-robotframework-runner-maven-plugin:1.0.0-
↪RELEASE:list -Dtf.disableMetadata=true
```

1.2 Robot Framework tests running

This Mojo enables one to run a selection of, or all possible, Robot Framework tests and report their execution. In order to do so one only needs to run the following command :

```
mvn org.squashtest.ta.galaxia:squash-tf-robotframework-runner-maven-plugin:1.0.0-
↪RELEASE:run
```

- `mvn` : launch Maven
- `org.squashtest.ta.galaxia:squash-tf-robotframework-runner-maven-plugin:1.0.0-RELEASE:run` : the actual running Mojo provided by **Squash TF** Robot Framework Runner.

By default the whole collection of tests available in the project will be executed. A summary of the execution is reported and available at `target/squashTA/html-reports/squash-ta-report.html`.

If one wants to only run a subset of possible test one can provide a list of :

- tests cases with the maven property "tf.test.suite"

  ```
  mvn org.squashtest.ta.galaxia:squash-tf-robotframework-runner-maven-plugin:1.0.0-
  ↪RELEASE:run -Dtf.test.suite="MyTest/Ihm/Add data;MyTest/Rest/Clean data"
  ```

  Warning: Be careful, the full qualified test case name is case sensitive. To prevent problems use the names provided by the `list` goal

- tests suites (.robot file) with the maven property "tf.robot.testSuite"

  ```
  mvn org.squashtest.ta.galaxia:squash-tf-robotframework-runner-maven-plugin:1.0.0-
  ↪RELEASE:run -Dtf.robot.testSuite="myTest/ihm.robot;myTest/rest.robot"
  ```

Note: These two properties are mutually exclusive. You can only use one of them in a command line.

For both of this maven property two mechanism are possible:

- Mimic TM-TF link and provide a list of selected test via a Json file. In this scenario the parameter should be given the value `{file:testsuite.json}` and the testsuite.json file should be put at the root of the project.
- Provide a CSV like line, where qualified tests names are listed separated by semicolons

Warning: If there are metadata syntax errors in the running test script(s), **warning** message(s) will be displayed in the console. (See *Metadata in Robot Framework Runner* for more information about Metadata syntax conventions)

1.3 Robot Framework test - Metadata Checking

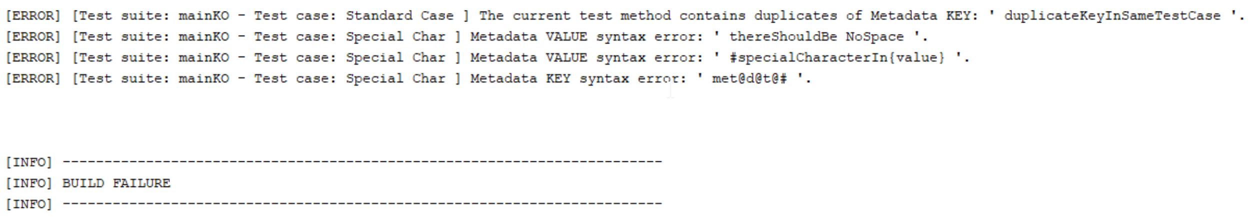
As the goal **"list"**, the goal *'check-metadata'* searches and checks if all metadata respect the conventions for writing and using Squash TF metadata (See *Metadata in Robot Framework Runner* for more information about Metadata syntax conventions). Contrary to goal **"list"**, there is no JSON file generated at the end of a successful **"check-metadata"** goal.

```
mvn org.squashtest.ta.galaxia:squash-tf-robotframework-runner-maven-plugin:1.0.0-
↪RELEASE:check-metadata
```

The goal will check through the project and collect all the metadata error(s).

If there are any error, the build will fail and a list of the incorrect metadata will be displayed. Otherwise, a **SUCCESS** result will be obtained.

The test suite and test case names are also displayed with the incorrect metadata.



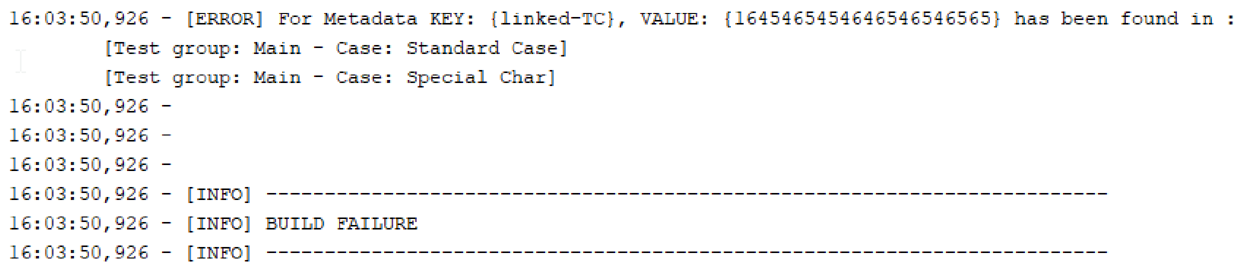
```
[ERROR] [Test suite: mainKO - Test case: Standard Case ] The current test method contains duplicates of Metadata KEY: ' duplicateKeyInSameTestCase '.
[ERROR] [Test suite: mainKO - Test case: Special Char ] Metadata VALUE syntax error: ' thereShouldBe NoSpace '.
[ERROR] [Test suite: mainKO - Test case: Special Char ] Metadata VALUE syntax error: ' #specialCharacterIn{value} '.
[ERROR] [Test suite: mainKO - Test case: Special Char ] Metadata KEY syntax error: ' met@d@t@# '.



[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
```

1.3.1 'check-metadata' goal with Unicity checking

In addition to the normal syntax checking, you can insert the *tf.metadata.check* property after the goal "check-metadata" to check the unicity of each Metadata Key - Value pair.

```
mvn org.squashtest.ta.galaxia:squash-tf-robotframework-runner-maven-plugin:1.0.0-
↪RELEASE:check-metadata -Dtf.metadata.check=[valueUnicity]
```

If there is any metadata Key-Value duplicate in the project(even if the syntax is OK), a **FAILURE** result will be obtained.



```
16:03:50,926 - [ERROR] For Metadata KEY: {linked-TC}, VALUE: {16454654546465465465} has been found in :
        [Test group: Main - Case: Standard Case]
        [Test group: Main - Case: Special Char]
16:03:50,926 -
16:03:50,926 -
16:03:50,926 -
16:03:50,926 - [INFO] ------------------------------------------------------------------------
16:03:50,926 - [INFO] BUILD FAILURE
16:03:50,926 - [INFO] ------------------------------------------------------------------------
```

'check-metadata' goal with Unicity checking for specific Keys

You can even check the unicity of each metadata Key - Value pair with just some specific Keys by inserting the second property *tf.metadata.check.key* after the first one mentioned above.

```
mvn clean compile test-compile org.squashtest.ta.galaxia:squash-tf-junit-runner-maven-
↪plugin:1.1.0-RELEASE:check-metadata -Dtf.metadata.check=[valueUnicity] -Dtf.
↪metadata.check.keys=[xxx,yyy,zzz]
```

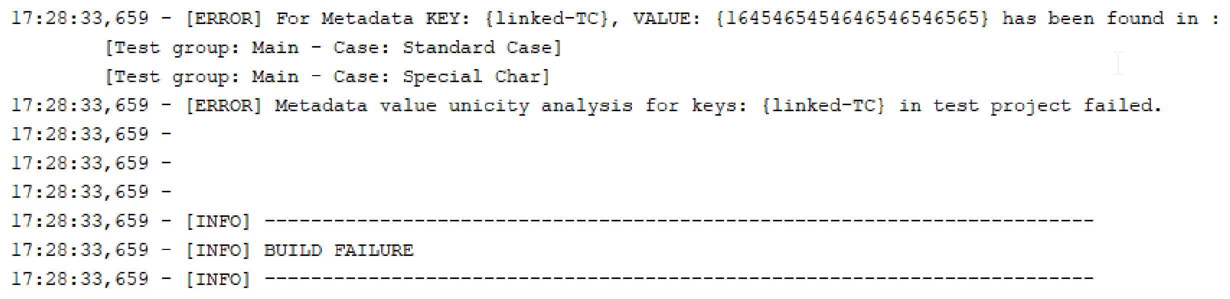
Important: In the bracket, the key list MUST be a string of characters composed by concatenation from 1 to n keys separated by commas: -Dtf.metadata.check.keys=[xxx,yyy,zzz]

If the list is surrounded by double quotes, spaces are allowed: -Dtf.metadata.check.keys="[xxx, yyy, zzz]"

It is NOT allowed to have two commas without any key OR only spaces/tabulations between them. (ex: -Dtf.metadata.check.keys="[xxx, ,yyy,,zzz]")

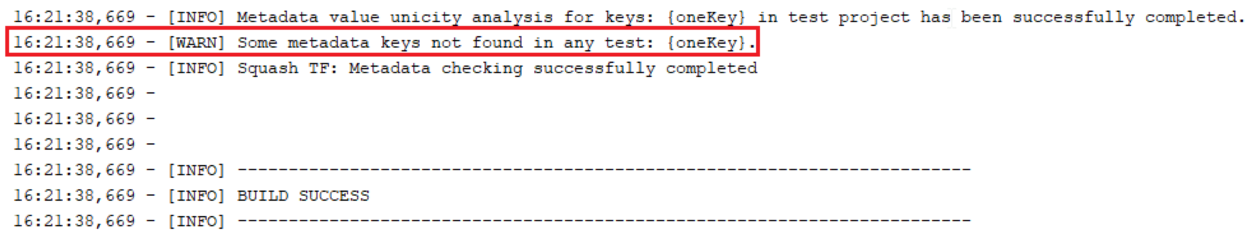
Key list is NOT allowed to be either uninitiated or empty. (ex: -Dtf.metadata.check.keys= OR -Dtf.metadata.check.keys=[])

For each searched metadata key, if there is any metadata Key-Value duplicate in the project, a **FAILURE** result will be obtained.



```
17:28:33,659 - [ERROR] For Metadata KEY: {linked-TC}, VALUE: {1645465454646546546565} has been found in :
        [Test group: Main - Case: Standard Case]
        [Test group: Main - Case: Special Char]
17:28:33,659 - [ERROR] Metadata value unicity analysis for keys: {linked-TC} in test project failed.
17:28:33,659 -
17:28:33,659 -
17:28:33,659 -
17:28:33,659 - [INFO] ------------------------------------------------------------------------
17:28:33,659 - [INFO] BUILD FAILURE
17:28:33,659 - [INFO] ------------------------------------------------------------------------
```

Note: If searched metadata key(s) are not found in any Test files, a **WARNING** message will be raised in the console.



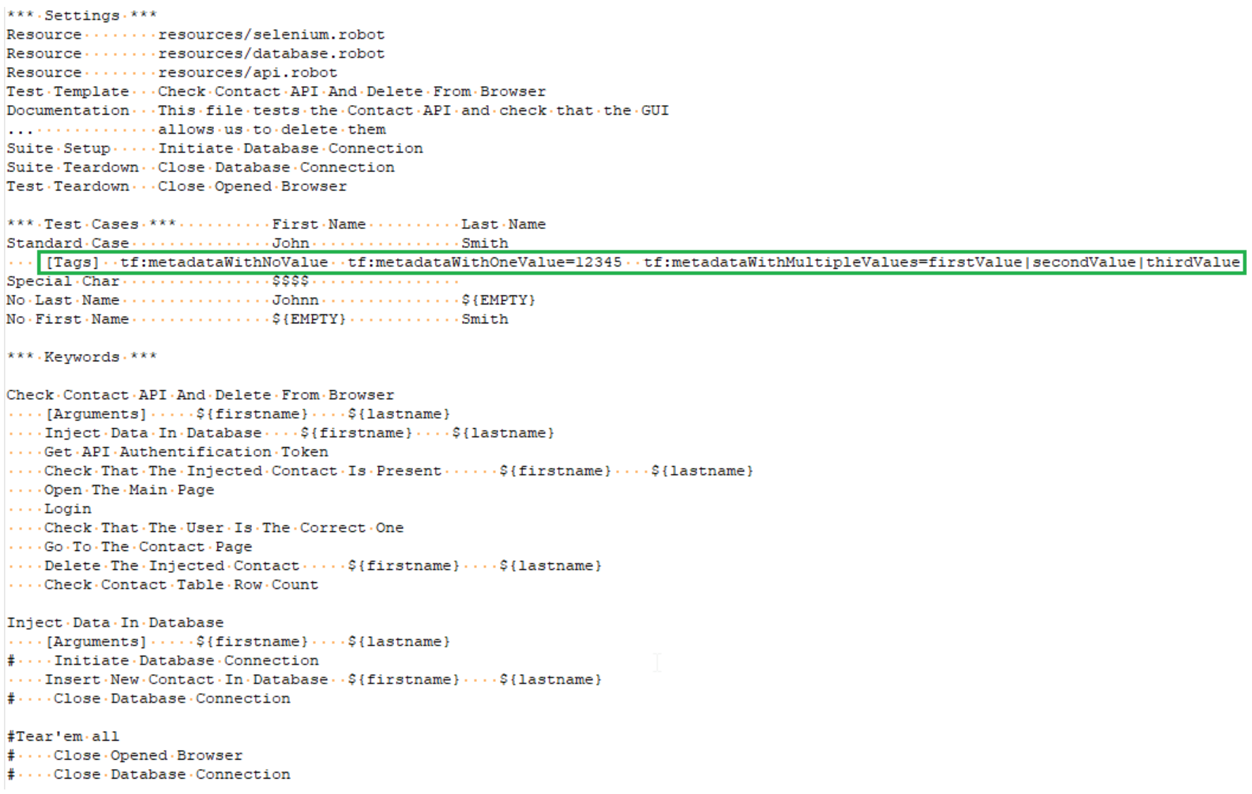
```
16:21:38,669 - [INFO] Metadata value unicity analysis for keys: {oneKey} in test project has been successfully completed.
16:21:38,669 - [WARN] Some metadata keys not found in any test: {oneKey}.
16:21:38,669 - [INFO] Squash TF: Metadata checking successfully completed
16:21:38,669 -
16:21:38,669 -
16:21:38,669 -
16:21:38,669 - [INFO] ------------------------------------------------------------------------
16:21:38,669 - [INFO] BUILD SUCCESS
16:21:38,669 - [INFO] ------------------------------------------------------------------------
```

CHAPTER 2

Metadata in Robot Framework Runner

In your **.robot** file, you can insert Squash Metadata into test cases via tags.

For example:



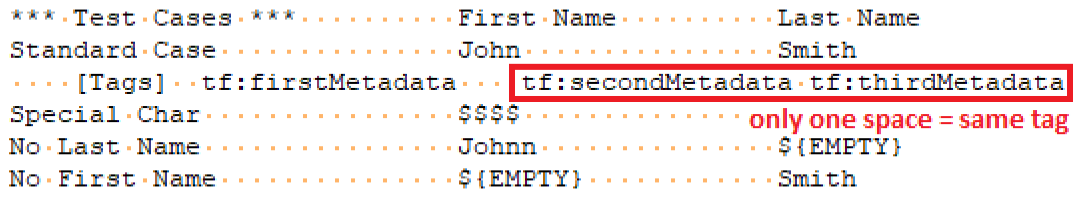
```
*** Settings ***
Resource        resources/selenium.robot
Resource        resources/database.robot
Resource        resources/api.robot
Test Template   Check Contact API And Delete From Browser
Documentation   This file tests the Contact API and check that the GUI
...             allows us to delete them
Suite Setup     Initiate Database Connection
Suite Teardown  Close Database Connection
Test Teardown   Close Opened Browser

*** Test Cases ***          First Name          Last Name
Standard Case               John                Smith
    [Tags]  tf:metadataWithNoValue  tf:metadataWithOneValue=12345  tf:metadataWithMultipleValues=firstValue|secondValue|thirdValue
Special Char                $$$$
No Last Name                Johnn               ${EMPTY}
No First Name               ${EMPTY}            Smith

*** Keywords ***

Check Contact API And Delete From Browser
    [Arguments]     ${firstname}    ${lastname}
    Inject Data In Database    ${firstname}    ${lastname}
    Get API Authentification Token
    Check That The Injected Contact Is Present      ${firstname}    ${lastname}
    Open The Main Page
    Login
    Check That The User Is The Correct One
    Go To The Contact Page
    Delete The Injected Contact     ${firstname}    ${lastname}
    Check Contact Table Row Count

Inject Data In Database
    [Arguments]     ${firstname}    ${lastname}
#    Initiate Database Connection
    Insert New Contact In Database  ${firstname}    ${lastname}
#    Close Database Connection

#Tear'em all
#    Close Opened Browser
#    Close Database Connection
```

2.1 Metadata syntax conventions

In order to work properly, the following syntax rules have to be followed.

2.1.1 Spaces

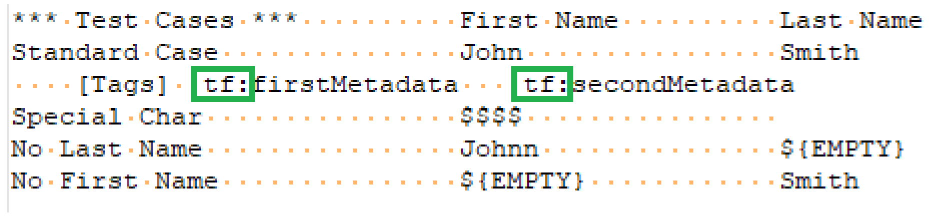
In a Robot script, all tags must at least have **two spaces** between each other to be considered as a new tag. If you put only one space, it will be considered as the same tag.



```
*** Test Cases ***            First Name          Last Name
Standard Case                 John                Smith
    [Tags]  tf:firstMetadata   tf:secondMetadata tf:thirdMetadata
Special Char                  $$$$
No Last Name                  Johnn               ${EMPTY}
No First Name                 ${EMPTY}            Smith
```

2.1.2 Prefix

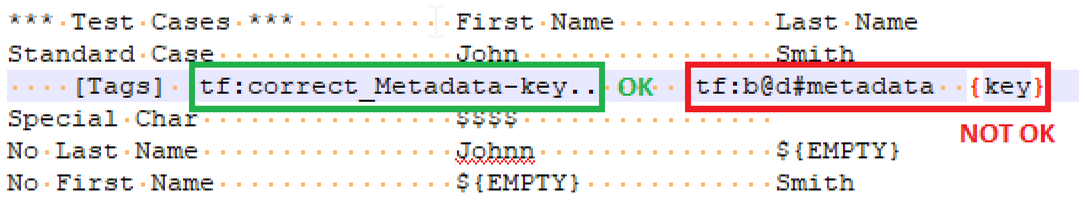
To be recognized as a metadata, you have to prefixed it with **"tf:"**.



```
*** Test Cases ***            First Name          Last Name
Standard Case                 John                Smith
    [Tags]  tf:firstMetadata   tf:secondMetadata
Special Char                  $$$$
No Last Name                  Johnn               ${EMPTY}
No First Name                 ${EMPTY}            Smith
```

2.1.3 Metadata Key

In a Metadata annotation, the **key** is mandatory. A metadata key MUST be ONE WORD which contains only *alphanumeric characters, dashes, underscores and dots*.



```
*** Test Cases ***            First Name          Last Name
Standard Case                 John                Smith
    [Tags]  tf:correct_Metadata-key..  OK  tf:b@d#metadata  {key}
Special Char                  $$$$
No Last Name                  Johnn               ${EMPTY}
No First Name                 ${EMPTY}            Smith
```

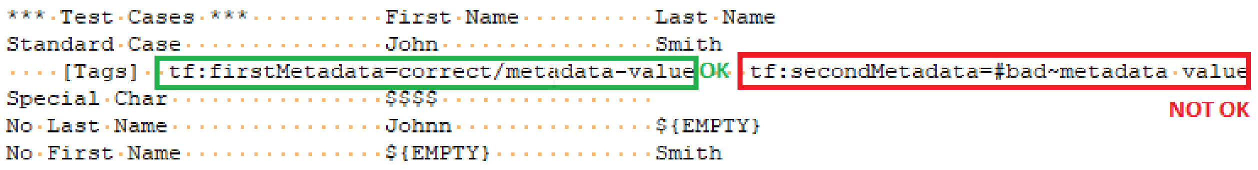
Important: Metadata key is **case insensitive** and must be **unique** in a test case.

The same metadata key can be found in different test cases **but values can't be the same**. This is not a problem if the metadata key has no value.

2.1.4 Metadata Value

The **value** is optional. A metadata value MUST be ONE WORD which contains only *alphanumeric characters, dashes, slashes, underscores and dots*.

You must separate the metada key and value with an "=" symbol.



```
***·Test·Cases·***··········First·Name··········Last·Name
Standard·Case···············John················Smith
····[Tags]··tf:firstMetadata=correct/metadata-value··tf:secondMetadata=#bad~metadata·value
Special·Char················$$$$················
No·Last·Name················Johnn···············${EMPTY}
No·First·Name···············${EMPTY}············Smith
```

Important: Metadata value is **case sensitive** and must be assigned to a metadata key.

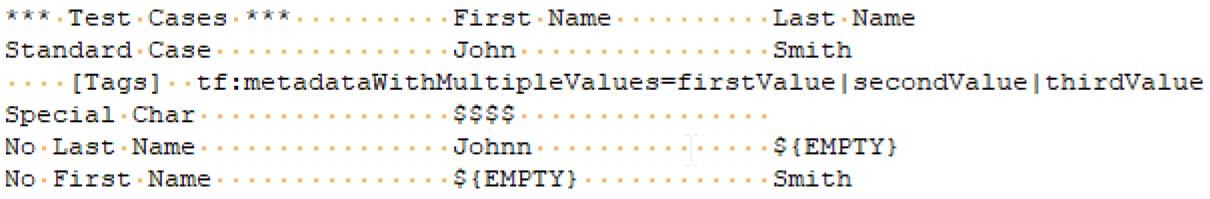
If you duplicate two tags with exactly the same key and value, Robot Framework will automatically ignore the second tag with the same value.

Furthermore, if you put two metadata **in the same test case** with the same key but with different values, our runner will inform you that there is a duplicate key.

2.1.5 Metadata with multiple values

It is possible to have many metadata values associated to a same key.

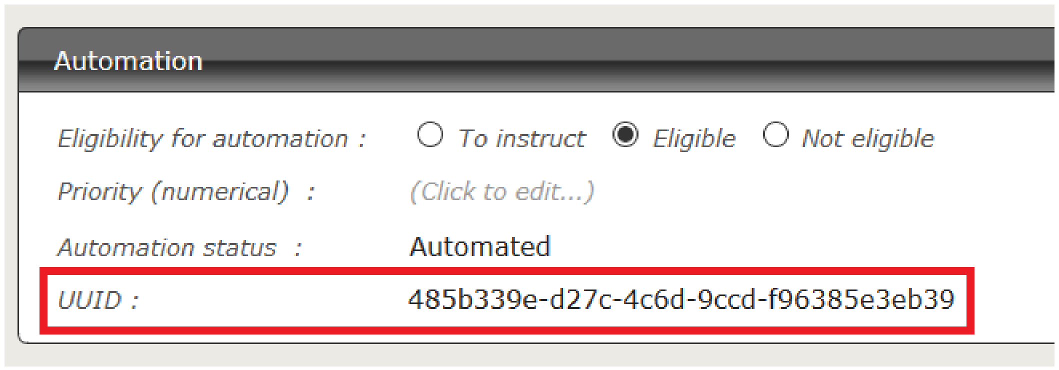
To separate the different values, you must use a "|" symbol (Press AltGr and 6).



```
***·Test·Cases·***··········First·Name··········Last·Name
Standard·Case···············John················Smith
····[Tags]··tf:metadataWithMultipleValues=firstValue|secondValue|thirdValue
Special·Char················$$$$················
No·Last·Name················Johnn···············${EMPTY}
No·First·Name···············${EMPTY}············Smith
```

2.2 Use metadata for TM - TF autolink

TF metadata handles the TM - TF autolink. Autolink is a feature to ease the link between a TM test case and a test automation script. On TM side, a UUID is now provided (when the workflow is activated):



Automation

Eligibility for automation : ○ To instruct [✓ Eligible] ○ Not eligible

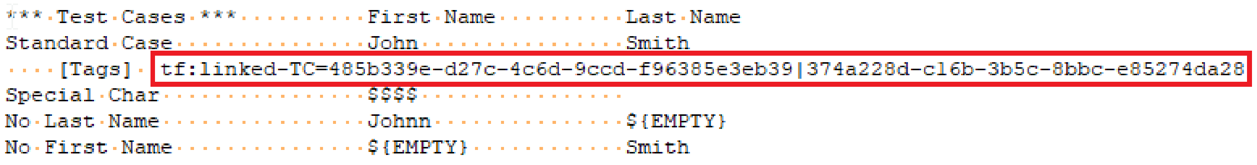
Priority (numerical) : (Click to edit...)

Automation status : Automated

UUID : 485b339e-d27c-4c6d-9ccd-f96385e3eb39

This UUID is used as an identifier.

In your automation test, add a TF Metadata which key is `linked-TC` and value is the UUID from the corresponding TM test case. As you can see in the example below, it's possible to link many TM test cases to the same automation test (two UUID are set in the "value" separated by a "|"):



```
*** Test Cases ***          First Name          Last Name
Standard Case               John                Smith
    [Tags]  tf:linked-TC=485b339e-d27c-4c6d-9ccd-f96385e3eb39|374a228d-c16b-3b5c-8bbc-e85274da28
Special Char                $$$$
No Last Name                Johnn               ${EMPTY}
No First Name               ${EMPTY}            Smith
```

CHAPTER 3

TF Param Service

This component provided by the `squash-tf-services` library allows you to retrieve the values of the parameters transmitted to the runner through the **json** test suite file (typically provided by **Squash TM**) in order to use them in your robotframework tests. All keywords defined by the library allow the caller to define default values in the test, so that it may be run outside of the runner.

3.1 Configuration

In order to use the TFParamService keywords in your test suite, you have to

- install the python library. We assume here that the pip python package manager is installed.

  ```
  python -m pip install squash-tf-services
  ```

  Make sure that this command has been executed in all environement where the tests will be run, in the relevant python environement. If you use a symlink and path setting in a linux environment to use python3 as your 'python' binary, make sure this setting is in effect when you run the install command.

- Import the keyword library by using the following Library statement in your test :

  ```
  Library squash_tf.TFParamService
  ```

3.2 Using parameter values in your tests

To retrieve parameter values in your tests, use one of the three defined keywords in the following way :

```
${value}= Get Param    <key>    [<optional_default_value>]
```

You may omit the optional default value if `None` is manageable in your test. Otherwise, the second parameter will define a default value used if no parameter value is defined for your key. This will happen if :

- The test is executed outside of the runner

- There is no parameter value for the key you have used (for exemple, no value for this test case)

3.2.1 Known limitations

If you use python2, encoding will not be properly managed in parameter values. This problem does not exist if you run the same tests using python3. As python2 will reached its end of life on next january (official EOL date 2020/01/01), we recommend using python3.

3.3 Keywords

${value}=	Get Global Param	key	
${value}=	Get Global Param	key	default_value

  This keyword retrieves the value of a parameter defined for all tests executed in this run. If there is a global value mapped on this key, the `value` argument is assigned with it, otherwise it is assigned with the default value. If no default value was given, `value` becomes `None`. Ex :

  ```
  ${parameter}=    Get Global Param    target_base_url    http:localhost:8080/sut/
  ```

${value}=	Get Test Param	key	
${value}=	Get Test Param	key	default_value

  This keyword retrieves the value of a parameter defined for one execution of a test case in this run. This means that the run, as built through the json test suite file build by Squash TM, may schedule several execution of the same test case with different value for the parameter. If there is a value mapped on this key for the specific test case run, the `value` argument is assigned with it,otherwise it is assigned with the default value. If no default value was given, `value` becomes `None`. Ex :

  ```
  ${login}=    Get Test Param    TC_login    test_user
  ```

${value}=	Get Param	key	
${value}=	Get Param	key	default_value

  This keyword combines both scopes. If a parameter value is mapped on this key for this test case run, then the `value` argument is assigned with this value. If not, global parameters are queried. If a global parameter value is mapped on the given key, then the `value` argument is assigned with this value. Otherwise, `value` becomes `None`. Ex :

  ```
  ${login}=    Get Param    login    test_user
  ```

3.4 Manually providing a .json file

If you want to manually provide a **.json** file, you need to add the following parameter on your runner command line **-Dtf.test.suite={file:path/to/json/FileName.json}**.

"path/to/json/FileName.json" must be the relative path of your **.json file** from the root of your project.

If the **.json file** is located directly at the root of your project, just type **-Dtf.test.suite={file:FileName.json}**

For example :

```
mvn org.squashtest.ta.galaxia:squash-tf-robotframework-runner-maven-plugin:1.0.0-
↪RELEASE:run -Dtf.test.suite={file:testSuite.json}
```

```
{
    "test": [{
            "id": "39",
            "script": "My Test Suite/First Test Case",
            "param": {
                "TC_REFERENCE": "",
                "TC_CUF_CUF_CUSTOM": "true",
                "TC_UUID": "3a7099ff-ab59-4e99-b21d-07e7d71d1ed5"
            }
        }, {
            "id": "40",
            "script": "My Test Suite/Second Test Case",
            "param": {
                "TC_REFERENCE": "",
                "DS_name": "Bertrand",
                "DSNAME": "dataset1",
                "TC_CUF_CUF_CUSTOM": "true",
                "DS_age": "41",
                "TC_UUID": "adec6164-5dec-4c8c-a0ae-d836370d519b"
            }
        }, {
            "id": "41",
            "script": "My Test Suite/Second Test Case",
            "param": {
                "TC_REFERENCE": "",
                "DS_name": "Damien",
                "DSNAME": "dataset2",
                "TC_CUF_CUF_CUSTOM": "true",
                "DS_age": "undefined",
                "TC_UUID": "adec6164-5dec-4c8c-a0ae-a036bf0d519b"
            }
        }],
    "param": {
        "globalParamSection": "This is global param section",
        "user": "foo",
        "ow_ner.Na-me": "bar",
    }
}
```

CHAPTER 4

Overview

The **Squash T**est **F**actory (**Squash TF**) Robot Framework Runner aims to provide a seamless integration to our ecosystem when automated test are implemented using Robot Framework as the underlying test framework.

As a **Squash TF** runner its main goal is twofold :

- List in JavaScript Object Notation (Json) format the available implemented tests
- Run a selection (that can include all available tests) and report the execution. In the case where the execution order originates from **Squash T**est **M**anagement (**Squash TM**), test status and report are sent back to **Squash TM**.

CHAPTER 5

Prerequisites

For the Squash TF Robot Framework runner to work, you should have installed on your execution environment (in addition to the recommendation applicable to all Squash TF runners) :

- Python 2 or 3 (the "python" command should be in your be in your path) To use the runner python3 under linux, you need to create a symlink named python that points to the to the python3 executable and put the directory where this link is located at the beginning of the PATH variable in your execution environement. In jenkins, this can be done in the dedicated node configuration.
- Robot Framework (the version you want)
- all robot external libraries requisite by your test project

Note: The command :

```
python -m robot --version
```

should succeed in order to Squash TF Robot Framework runner works.