**USER MANUAL FOR EXECUTING SQUASH DSLs**

**IN INTELLIJ COMMUNITY IDE**

August 2019

Version 1.0.8



Document Revisions

|  |  |  |
| --- | --- | --- |
| Date | Version Number | Document Changes |
| 01/04/2019 | 0.1 | Initial Draft |
| 05/04/2019 | 0.2 | Need Image update for Test DSL Autocompletion feature |
| 08/04/2019 | 0.3 | Ready for Validation |
| 23/04/2019 | 0.4 | Insert “Squash Build Configuration” section |
| 26/08/2019 | 0.5 | Replace all “command” words with “basic instruction”Insert some new syntax rule notes for each DSLInsert METADATA section in Squash Test scriptUpdate “Related documents” link to: https://squash-tf.readthedocs.io/en/latest/Remove the Limitation saying that we can only write characters matched this regex: [a-zA-Z0-9\/.\_-] for a value.Modifying the section 4.3, 4.4, 4.5Modifying the chapter namesupdate new behaviors for Macro auto-completion mechanism |
| 27/08/2019 | 0.6 | Update the syntax rule for the Macro value |
| 13/09/2019 | 0.7 | Update “**Create configurations for Squash TF builds**” section |
| 18/09/2019 | 0.8 | Update “**list**” and “**check-metadata**” goalsUpdate auto-completion “known limitations” |
|  |  |  |
|  |  |  |

Table of Contents

1 Introduction 3

1.1 Scope and Purpose 3

1.2 Squash DSL background - Related documents 3

1.2.1 Macro DSL 3

1.2.2 Test DSL 4

1.2.3 Related documents 5

1.3 Overview – Squash TF plugin features 5

2 Plugin installation into ICI 6

2.1 Install plugin from disk 6

2.2 Install plugin from IntelliJ marketplace 6

3 Create a Squash project in ICI 7

4 ICI Squash TF plugin features 9

4.1 Squash DSL file template with descriptions 9

4.2 Squash DSL syntax coloration 10

4.2.1 Macro DSL 10

4.2.2 Test DSL 10

4.3 Squash DSL Annotation 11

4.3.1 Macro DSL 11

4.3.2 Test DSL 13

4.4 Squash DSL autocompletion 15

4.4.1 Macro DSL 15

4.4.2 Test DSL 18

4.5 Squash DSL line marker/navigation 19

# Introduction

## Scope and Purpose

This document is a User Guide document for use by Squash projects. It provides guidance, explanations and illustrations which is intended to assist the writing of the Squash Domain-Specific Languages (DSLs) in the IntelliJ Community Integrated Development Environment (ICI).

It is also useful background reading for anyone involved in developing, monitoring or executing a Squash Test project/system.

## Squash DSL background - Related documents

There are two Squash DSLs: **Macro DSL** used in Squash Macro file and **Test DSL** used in Squash Test script.

### Macro DSL

* Definition: Macro DSL is a specific language that constitutes a Squash Macro file (.macro extension).
* *Squash Macro file* is used in a Squash project (in ‘*shortcuts’* folder) to define the content of a Macro line which can be used in a Squash Macro file or a Squash Test script (.ta extension).

|  |  |
| --- | --- |
| *Figure 1 -* *Structure of a Squash project** Squash Macro file
* Squash Test script
 |  |

* There are 3 main components in a Squash Macro file:
	+ *Signature*: the syntax definition of a Macro (note: while the **name** of a macro file can be different from its **signature** defined in its content, the identifier of a macro used for its invocation is the **signature**). It must be the first line of the Macro file.
	+ *Separator*: separating the signature and the body of a macro file at must be placed in the next line right after the macro signature.
	+ *Body*: a set of Squash instructions that will be executed by the Squash engine upon their appearance order
		- *Squash instruction*: either a basic instruction line or a Macro line

|  |
| --- |
| *Figure 2 - Structure of a Squash Macro file* |
| * Signature
* Separator
* Body
* Instruction line
* Macro line
 |  |

* *Macro line* is the shortcut of a Squash instruction set which can be called from a Squash Test script or a Squash Macro file. Its syntax must respect one signature structure defined in a Macro file. For example:
	+ # ASSERT XML xml\_path IS {valid} USING SCHEMA {{xsd\_path}}

Yellow: macro keywords

Green: user input values/file paths

Note: For a macro line to be recognized, the syntax and position of its keywords must be completely respected. However, there are no specific rules for the file path or input values.

* *Basic instruction line* is one of these 6 templates:
	+ DEFINE $(raw data) AS {nameInTheContext}
	+ LOAD {path\_To\_Resource} [ FROM {resourceLibrary} ] [ AS {nameInTheContext} ]
	+ CONVERT {resourceToConvert} TO <Category>( <Conv> ) [ USING {config} ] AS {convertedResource}
	+ EXECUTE <Cmd> WITH {resource} [ ON {target} ] [ USING {config} ] AS {result}
	+ ASSERT {resourceToTest} ( IS | HAS | DOES ) <Asr> [ ( WITH | THAN | THE ) {expectedResult} ] [ USING {config} ]
	+ VERIFY {resourceToTest} ( IS | HAS | DOES ) <Asr> [ ( WITH | THAN | THE ) {expectedResult} ] [ USING {config} ]

[...] : optional

(…|…) : choose one in group

Red: instruction head keyword

Blue: instruction keyword

Green: user input value/file path

Rose: engine component identifier

* *Comment line* (starting with “//”) and empty line (line containing nothing or only spaces/tabulations) can be inserted **anywhere after the Separator**.

###

### Test DSL

* Definition: Test DSL is a specific language that constitutes a Squash Test script (.ta extension).
* *Squash Test script* is used in a Squash project (in ‘*tests’* folder or its subfolder) to define the content of a Squash test case (see *Figure 1 - Structure of a Squash project*).
* There are 1 informative section and 3 execution sections (phases) in a Squash Test script:
	+ METADATA (informative section) (optional)
	+ SETUP (optional)
	+ TEST (required)
	+ TEARDOWN (optional)
* In each section is a set of Squash instructions that will be executed by the Squash engine upon their appearance order.
* *Squash instruction*: either a Basic instruction line or a Macro line (see [Macro DSL section](#_Macro_DSL))
* *Comment line* (starting with “//”) and empty line (line containing nothing or only spaces/tabulations) can be inserted anywhere.

|  |
| --- |
| *Figure 3 - Structure of a Squash Test script* |
| * METADATA section
* SETUP phase
* TEST phase
* TEARDOWN phase
 |  |

### Related documents

A wiki for more detailed information about SQUASH DSLs/project and their components can be found [here](https://squash-tf.readthedocs.io/en/latest/).

## Overview – Squash TF plugin features

To facilitate the employing of Squash DSL components during the creation/modification of a Test script or a Macro file that is opened in an ICI, the Squash TF plugin has been integrated into ICI (version 2018.3.2 or later) to provide the following features:

1. Squash DSL file template with descriptions
2. Squash DSL syntax highlighting
3. Squash DSL syntax validation
4. Squash DSL component autocompletion
5. Squash DSL line marker/navigation

# Plugin installation into ICI

Requirement:

* IntelliJ Community IDE (version 2018.3.2 or later recommended)
* JAVA JDK 1.8
* MAVEN 3.5.0

## Install plugin from disk

The ICI Squash TF plugin can be downloaded here.

Follow the instructions [here](https://www.jetbrains.com/help/idea/managing-plugins.html#install_plugin_from_disk) to install the plugin archived file to user ICI:



*Figure 4 – Install plugin from disk for ICI*

## Install plugin from IntelliJ marketplace

To be updated in next versions

# Create a Squash project in ICI

For anyone who uses ICI for the first time, when launching ICI after the installation, a window will appear to ask for creating or importing a project (Figure 5).

 *Figure 5 – ICI first launching interface Figure 6 – Choosing project template*

By clicking on “Create New Project” option, a new window appears for choosing a project template (Figure 6). Choose “**MAVEN**” as template for creating a Squash project.



*Figure 7 – Maven archetype configuration for Squash project*

Project SDK: JAVA 1.8 is recommended. Check: “Create from archetype” option and Enter the following values:

* Archetype Group ID: org.squashtest.ta
* Archetype Artifact ID: squash-ta-project-archetype
* Archetype Version: the last Squash-TA version (ex: 1.13.0-RELEASE)
* Repository URL: http://repo.squashtest.org/maven2/releases/

User can then define the Maven properties: GroupId, ArtifactId, Version, indicate the MAVEN home directory (version 3.5.0 is recommended) as well as the name and location for his/her project.

Typically, with a Maven BUILD SUCCESS, the Squash project is ready for experimenting all the ICI Squash TF plugin features.

# Create configurations for Squash TF builds in ICI

To create/insert a build for a project in ICI, click on the Build Configuration option in the right-hand part of the ICI and choose **Add/Edit Configurations**… and choose the **MAVEN** options**.**



## Run all tests

In **Parameters** section > **Command line** option, insert this command:

**squash-ta:run -Dlog4j.configurationFile=log4j2.xml**

****

Note: Don’t forget to indicate the location for **Maven home directory** option in **General** section (version 3.5.0 recommended) !

## Run all tests DEBUG

In **Parameters** section > **Command line** option, insert this command:

**squash-ta:run -X -Dlog4j.configurationFile=log4j2.xml**

## Run selected test(s)

In **Parameters** section > **Command line** option, insert this command:

**squash-ta:run -Dlog4j.configurationFile=log4j2.xml -Dta.test.suite=”insert the Test file path manually”**

For example: squash-ta:run -Dlog4j.configurationFile=log4j2.xml -Dta.test.suite=simple.ta

Note: The file path can be absolute or relative to the project “src\squashTA\tests” folder.

## Run selected test(s) DEBUG

In **Parameters** section > **Command line** option, insert this command:

**squash-ta:run -X -Dlog4j.configurationFile=log4j2.xml -Dta.test.suite=”insert the Test file path manually”**

## Run test list

In **Parameters** section > **Command line** option, insert this command:

**squash-ta:run -Dlog4j.configurationFile=log4j2.xml -Dta.test.suite=”insert the list of Test file paths manually, separated by comma (,)”**

## List tests

In **Parameters** section > **Command line** option, insert this command:

**squash-ta:list -Dlog4j.configurationFile=log4j2.xml [-DdisableMetadata=true]**

## Check-metadata

In **Parameters** section > **Command line** option, insert this command:

**squash-ta:check-metadata -Dlog4j.configurationFile=log4j2.xml**

**[-Dtf.metadata.check=[valueUnicity]] [-Dtf.metadata.check.keys=[KEY,missingOne,abc,123]]**

# ICI Squash TF plugin features

## Squash DSL file template with descriptions

The Squash TF plugin provides in ICI two DSL templates for Squash users to have a look at how a Squash Macro file or Test script is constituted. It presents also the name, the position as well as the color of every component of each DSL.

In File menu, choose Settings…🡪Editor🡪Color Scheme: then choose Squash Macro File option to view the Squash Macro File template.



*Figure 9 – Squash Macro File template*

* By clicking on a component name listed in the upper window, user can observe its color description in the right column and its position indicated in the lower Squash Macro file sample.
* Clicking on an element in the file sample also gives the user the corresponding information in the upper and right windows.

The same behaviors are also provided in the Squash Test Script template.

## Squash DSL syntax highlighting

As observed in “[Squash DSL template](#_Squash_DSL_file)” section, each Squash DSL owns a specific list of components. In ICI, each of these latest is identified depending on its position in the file content and is respectively assigned to a unique color (read [here](#_Squash_DSL_background) for more detailed information).

### Macro DSL



### Test DSL



## Squash DSL Validation

Also integrated in the ICI by Squash TF plugin is the real-time-checking capacity. In fact, whilst a Squash DSL file is opened and being composed, the plugin checks its current properties/content and creates an Annotation if needed.

### Macro DSL

* If a Macro file is not in the project “src/squashTA/shortcuts” folder, its whole content will be highlighted with an Error Annotation. Furthermore, all other features provided by the plugin such as syntax checking, auto-completion… will NOT be correctly supported.



* A macro file must control ONE (and only ONE) macro signature and ONE (and only ONE) separator. In a Squash macro file, the signature must be the first line followed by the separator. Otherwise it will invoke an Error Annotation. For example:









* If the current working macro signature template has already been used (either in the Squash framework or in user working project “shortcuts” folder), an Error Annotation will appear.



* In the macro file body, there are 3 different kinds of lines: comment line, basic instruction line and macro line. While there is no rule for a comment line except for that it must start with a “//”, a [macro line](#macro_line) whose keyword template doesn’t match any Macro signature (either in the Squash framework or in user working project “shortcuts” folder) will trigger an Error Annotation.



* If a macro line keyword template matches more than one Macro signature, a Warning Annotation will be raised.



* If a [basic instruction line](#command_line) doesn’t match any templates, an Error Annotation together with a tooltip to propose the 6 Squash basic instruction templates will be proposed.



* If a basic instruction line matches a template but contains undefined built-in value(s), an Error Annotation will show.



### Test DSL

* If a Test file is not in the project “src/squashTA/tests” folder, its whole content will be highlighted with an Error Annotation. Furthermore, all other features provided by the plugin such as syntax checking, auto-completion… will NOT be correctly supported.



* A test file must control ONE (and only ONE) TEST phase. Other sections are optional but also at most 1 section for each. The right order must be: METADATA > SETUP > TEST > TEARDOWN.



* In each phase of the Squash Test script, there are 3 different kinds of lines: comment line, basic instruction line and macro line. While there is no rule for a comment line except for that it must start with a “//”, a [macro line](#macro_line) containing a keyword list that matches no existing Macro signature (either in the Squash framework or in the working project “shortcuts” folder) will trigger an Error Annotation.



* On the other hand, if a macro line matches more than one Macro signature, a Warning Annotation will be raised.



* If a [basic instruction line](#command_line) doesn’t match any templates, an Error Annotation together with a tooltip to propose the 6 Squash basic instruction templates will be proposed.



* If a basic instruction line contains undefined built-in value(s), an Error Annotation will appear.



## Squash DSL component autocompletion

One of the most effective ways to facilitate the usage of a DSL is importing an autocompletion (AC) system; and the Squash TF plugin does provide one in ICI. Based on each precise context that the user cursor is currently in, a list of appropriate Squash DSL components will be proposed when the “ctrl + space” combo-key is taped.

### Macro DSL

* If an AC is demanded while the current Squash Macro file does not have a macro signature followed by a separator yet, a proposal for the missing part(s) will consequently be raised. User can then modify the given template to obtain his/her desired Macro signature.







* AC in macro body: With a valid macro signature and separator, when an AC is asked in the macro body, depending on each context (*i.e.* the current content of the working line) is proposed a specific Squash DSL template/component.



* + As seen in the picture above, when an AC is made at the beginning of an empty line, a “#” will be proposed for starting a macro line. Moreover, 6 Squash basic instruction line templates as well as their separated HEAD KEYWORD are also provided to initiate a new basic instruction line.
	+ If the user chooses to start with a specific basic instruction HEAD KEYWORD and then asks for an AC at the following VALUE position, *a tooltip* be proposed to indicate that a value identifier or file path should be inserted. In next versions, real and appropriate values (selected by value type upon the current AC position) will be proposed.



* + As the first pair CMD HEAD KEYWORD – VALUE is complete, the plugin will then propose all CMD KEYWORDs of that basic instruction template which are not presented in the current line yet.



* + Like the CMD HEAD KEYWORD case, only a *tooltip* will be proposed if an AC is asked at the value position of a CMD KEYWORD.



* + If an AC is required at the position that waits for a Squash *engine component identifier*, a list of appropriate values will be raised.



* + When a line starting with “# ” is on an AC demand, all the available macro signature templates and their first fixed part component will be proposed. The ones coming from the Squash framework will be in **bold** style.



* + Note:
		- Values proposed in an AC now is just a *pseudo-value name* that is defined in the matched macro signature



### Test DSL

All features provided for an AC in a macro line and a basic instruction line of a Squash Test script are the same of a Squash Macro file. The only difference between the two Squash DSL AC is that in a Squash Test script, instead of providing ACs for signature/separator, a PHASE AC is added.

* + If the current Test script doesn’t have a METADATA section or a PHASE among SETUP, TEST and TEARDOWN phases, a suitable proposal will be invoked upon the cursor current position.









* + AC in Phase content: When a phase is well defined, all the ACs for a macro line or a basic instruction line are the same as in a Squash macro file. (see [here](#content_autocompletion))

## Squash DSL macro line marker/navigation

* When the macro signature of a Squash Macro file as well as a macro line in both two Squash DSL file is read by the ICI Squash DSL plugin, its Keyword template will be compared with all the existing macro signatures. If the template is found matched a Macro signature, a marker will be created for this current line.



* When the macro template comes from the Squash framework, a message is sent to user when his/her cursor is on the marker.



* In case of ‘project custom macro’, the marker proposes both a message and a navigation to the concerned Squash Macro File. Click on the link and you will be navigated to the macro file that defines this template.



* If more than one macro file in the project is found, a list of navigations will then be created.

