

---

# Squash Autom & Squash Devops

*Version 1.0.0-alpha1*

**squashtest**

**juin 28, 2021**



---

## Table des matières

---

<b>1</b>	<b>Squash AUTOM</b>	<b>3</b>
1.1	Guide d'Installation . . . . .	3
1.2	Pilotage de l'exécution de tests automatisés via un PEAC (Plan d'Exécution «as code») . . . . .	4
1.3	Pilotage de l'exécution de tests automatisés depuis Squash TM . . . . .	4
<b>2</b>	<b>Squash DEVOPS</b>	<b>23</b>
2.1	Guide d'Installation . . . . .	23
2.2	Appel au Squash Orchestrator depuis un pipeline Jenkins . . . . .	24
2.3	Récupération d'un plan d'exécution Squash TM depuis un PEAC . . . . .	26



**Squash AUTOM** est un ensemble de composants pour la gestion de l'exécution de vos tests automatisés.

**Squash DEVOPS** est un ensemble de composants pour l'intégration de l'exécution de vos tests fonctionnels automatisés à votre chaîne d'intégration continue.



### 1.1 Guide d'Installation

- *Squash Orchestrator*
- *Plugin Result Publisher pour Squash TM*

#### 1.1.1 Squash Orchestrator

**Squash Orchestrator** est disponible sous forme d'une image *Docker* sur DockerHub (*squashtest/squash-orchestrator :1.0.0.alpha2*).

Pour la méthode de déploiement, consulter la documentation de **Squash Orchestrator** (*Squash Orchestrator Documentation – 1.0.0.alpha2* version .pdf) téléchargeable depuis <https://www.squashtest.com/community-download>.

### 1.1.2 Plugin Result Publisher pour Squash TM

Le plugin existe en version **Community** (*squash.tm.rest.result.publisher.community-1.0.0.alpha2.jar*) librement téléchargeable ou **Premium** (*squash.tm.rest.result.publisher.premium-1.0.0.alpha2.jar*) accessible sur demande.

Pour l'installation, merci de vous reporter au protocole d'installation d'un plugin **Squash TM** (<https://sites.google.com/a/henix.fr/wiki-squash-tm/installation-and-exploitation-guide/2—installation-of-squash-tm/7—jira-plugin-in>).

<b>Avertissement :</b> Ce plugin est compatible avec une version <i>1.22.2.RELEASE</i> de <b>Squash TM</b> .
--

## 1.2 Pilotage de l'exécution de tests automatisés via un PEAC (Plan d'Exécution «as code»)

**Squash AUTOM** permet la rédaction de plans d'exécution dans un formalisme spécifique à **Squash Orchestrator**, les PEAC (Plan d'Exécution «as code»), pour orchestrer précisément l'exécution des tests automatisés en dehors d'un référentiel de test.

Retrouvez plus d'informations sur la rédaction d'un PEAC au sein de la documentation de **Squash Orchestrator** (*Squash Orchestrator Documentation – 1.0.0.alpha2* version .pdf) accessible depuis <https://www.squashtest.com/community-download>.

## 1.3 Pilotage de l'exécution de tests automatisés depuis Squash TM

- |  |
|--|
| <ul style="list-style-type: none"><li>— <i>Automatisation d'un cas de test Squash TM</i></li><li>— <i>Déclenchement d'un plan d'exécution depuis Squash TM</i></li><li>— <i>Remontées de résultats après exécution d'un plan d'exécution Squash TM</i></li></ul> |
|--|



### 1.3.1 Automatisation d'un cas de test Squash TM

**Note :** Cette page décrit les opérations communes à tous les frameworks de test supportés par cette version. Pour faciliter la navigation vous pouvez directement consulter les spécificités de l'automatisation de chaque technologie grâce aux liens suivants :

- [Cucumber](#)
- [Cypress](#)
- [JUnit](#)
- [Robot Framework](#)
- [SoapUI](#)

#### Sans utilisation de workflow d'automatisation Squash

Pour qu'un cas de test soit utilisable par **Squash Orchestrator**, il faut que le panneau *Automatisation* de l'onglet *Information* de la page de consultation d'un cas de test soit correctement renseigné :

Automatisation	
Technologie du test automatisé :	Robot Framework
URL du dépôt de code source :	https://my-scm/myrepo/my-repo (master)
Référence du test automatisé :	my-repo/test.robot#firstTestCase

- **Technologie du test automatisé :** Liste déroulante permettant de choisir la technologie utilisée pour exécuter le cas de test. Dans cette version, seuls les choix *Robot Framework*, *JUnit*, *Cucumber*, *Cypress* et *SoapUi* sont fonctionnels.
- **URL du dépôt de code source :** L'adresse du dépôt de source où se trouve le projet, tel que spécifié dans l'espace *Serveurs de partage de code source* de l'*Administration*.
- **Référence du test automatisé :** Correspond à l'emplacement du test automatisé dans le projet. Cette référence doit respecter un format propre à la technologie de test employée (voir [ici](#)).

#### Avec utilisation de workflow d'automatisation Squash

##### Cas de test classique

Pour qu'un cas de test soit utilisable par **Squash Orchestrator**, il doit être automatisé à partir de l'espace *Automatisation (Automaticien)* via trois colonnes à renseigner :

Tech. test auto.	URL du scm	Ref. test auto.
▼	▼	▼

- **Tech. test auto. :** Liste déroulante permettant de choisir la technologie utilisée pour exécuter le cas de test. Dans cette version, seuls les choix *Robot Framework*, *JUnit*, *Cucumber*, *Cypress* et *SoapUi* sont fonctionnels.
- **URL du SCM :** L'adresse du dépôt de source où se trouve le projet.
- **Ref. test auto. :** Correspond à l'emplacement du test automatisé dans le projet. Cette référence doit respecter un format propre à la technologie de test employée (voir [ici](#)).

### Cas de test BDD ou Gherkin

Les informations du panneau *Automatisation* sont remplis automatiquement lors de la transmission d'un script BDD ou Gherkin à un gestionnaire de code source distant. Ils sont également modifiables à tout moment par l'utilisateur.

---

### Exploitation de paramètres Squash TM

Au lancement d'un plan d'exécution **Squash TM** (via un PEAC ou directement depuis l'espace campagne), celui-ci transmet différentes informations sur les ITPI qu'il est possible d'exploiter dans un cas de test *Cucumber*, *Cypress*, ou *Robot Framework*. Les détails de cette fonctionnalité sont décrits dans la section correspondant à la technologie utilisée.

---

### Spécificités pour l'automatisation suivant le framework d'automatisation

#### Automatisation avec Cucumber

##### 1. Référence du test

---

**Note :** Dans cette version de **Squash AUTOM** il n'est pas possible de sélectionner un scénario spécifique dans un fichier `.feature` qui en contiendrait plusieurs : tous les scénarios du fichier sont donc exécutés ensemble. Le résultat de chaque exécution du cas de test **Squash TM** est calculé en prenant en compte les résultats individuels de chaque scénarios inclus dans le fichier lié :

- Si au moins un scénario est en statut *Error* (dans le cas d'un problème technique), l'exécution sera en statut *Blocked*.
  - Si au moins un scénario a échoué fonctionnellement et qu'aucun scénario n'est en statut *Error*, l'exécution sera en statut *Failed*.
  - Si tous les scénarios ont réussi, l'exécution sera en statut *Success*.
- 

Pour lier un cas de test **Squash TM** à un test automatisé *Cucumber*, le champ *Référence* du test automatisé du bloc *Automatisation* du cas de test doit avoir la forme suivante :

[1] / [2]

Avec :

- [1] : Nom du projet sur le dépôt de source.
- [2] : Chemin et nom du fichier de test *Cucumber* à partir de la racine du projet (avec son extension `.feature`).

##### 2. Nature des paramètres Squash TM exploitables

**Squash AUTOM** et **Squash DEVOPS** sont capables d'exploiter le nom d'un jeu de données Squash TM d'un cas de test comme valeur d'un tag à utiliser pour l'exécution d'un sous-ensemble spécifique d'une feature *Cucumber*.

Cette exploitation est possible par les composants en version **Community** ou **Premium**.

### 3. Utilisation de paramètres Squash TM

Il est possible lors de l'exécution d'un cas de test **Squash TM** automatisé avec *Cucumber* d'exploiter un nom de jeu de données **Squash TM** pour n'exécuter qu'un jeu de données particulier d'un scénario *Cucumber*.

Pour cela, il faut suivre les étapes suivantes :

- Renseigner des jeux de données dans l'onglet *Paramètres* du cas de test dans **Squash TM**.
- Créer au sein de son scénario *Cucumber* autant de tableaux exemple que de jeux de données et annoter ces tableaux d'un tag correspondant au nom d'un jeu de données **Squash TM**.
- Créer une seule ligne d'éléments dans chaque tableau exemple afin de fixer une valeur pour les différents paramètres du scénario.

Ci-dessous un exemple d'un fichier de test *Cucumber* et l'automatisation du cas de test **Squash TM** associé :

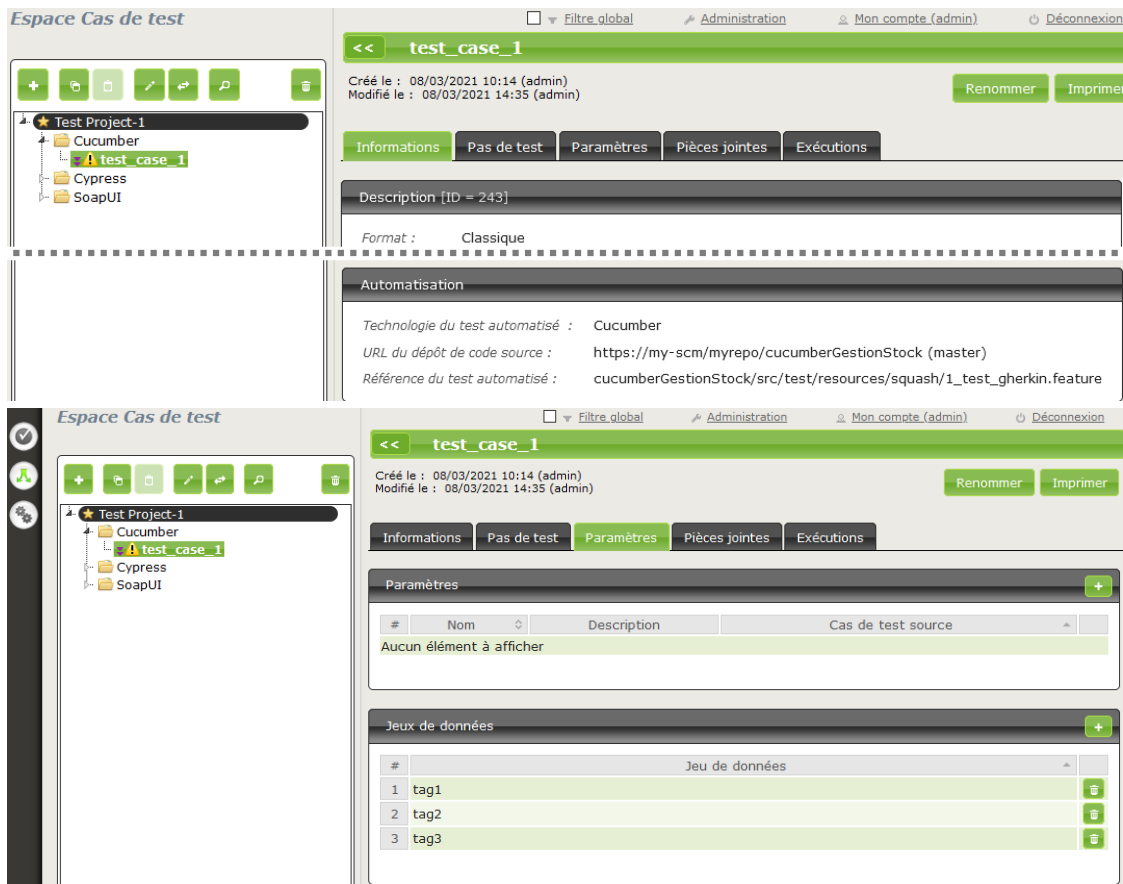
cucumberGestionStock / src / test / resources / squash / 1\_test\_gherkin.feature in master

```

<> Edit file    Preview changes

1  Feature: Gestion de stock
2
3  Plan du scénario: Ajout de stock
4    Etant donné que je dois ajouter des <élément>
5    Et que je détermine sa quantité
6    Quand je l'ajoute à mon stock
7    Alors je dois avoir un minimum de marchandises
8
9    @tag1
10   Exemples:
11     | élément |
12     | "Echelles" |
13
14   @tag2
15   Exemples:
16     | élément |
17     | "Coffres" |
18
19   @tag3
20   Exemples:
21     | élément |
22     | "Planches" |
23

```



## Automatisation avec Cypress

### 1. Référence du test

**Note :** Dans cette version de **Squash AUTOM** il n'est pas possible de sélectionner un test spécifique dans un fichier `.spec.js` qui en contiendrait plusieurs : tous les tests du fichier sont donc exécutés ensemble. Le résultat de chaque exécution du cas de test **Squash TM** est calculé en prenant en compte les résultats individuels de chaque test inclus dans le fichier lié :

- Si au moins un test est en statut *Error* (dans le cas d'un problème technique), l'exécution sera en statut *Blocked*.
- Si au moins un test a échoué fonctionnellement et qu'aucun test n'est en statut *Error*, l'exécution sera en statut *Failed*.
- Si tous les tests ont réussi, l'exécution sera en statut *Success*.

Pour lier un cas de test **Squash TM** à un test automatisé *Cypress*, le champ *Référence* du test automatisé du bloc *Automatisation* du cas de test doit avoir la forme suivante :

[1] / [2]

Avec :

- [1] : Nom du projet sur le dépôt de source.
- [2] : Chemin et nom du fichier de test *Cypress* à partir de la racine du projet (avec son extension `.spec.js`).

## 2. Nature des paramètres Squash TM exploitables

Les paramètres **Squash TM** exploitables dans un script *Cypress* vont différer suivant si vous utilisez les composants **Squash DEVOPS Community** ou **Squash DEVOPS Premium**.

Voici le tableau des paramètres exploitables :

Nature	Clé	Community	Premium
Nom du jeu de donnée	DSNAME	✓	✓
Paramètre d'un jeu de donnée	DS_[nom]	✓	✓
Référence du cas de test	TC_REF	✓	✓
CUF cas de test	TC_CUF_[code]	✓	✓
CUF itération	IT_CUF_[code]	✗	✓
CUF campagne	CPG_CUF_[code]	✗	✓
CUF suite de tests	TS_CUF_[code]	✗	✓

Légende :

- CUF : *Custom Field / Champ personnalisé*
- [code] : Valeur renseignée dans le champ "Code" d'un CUF
- [nom] : nom du paramètre tel que renseigné dans Squash TM

## 3. Utilisation de paramètres Squash TM

Il est possible lors de l'exécution d'un cas de test **Squash TM** automatisé avec *Cypress* d'exploiter des paramètres **Squash TM** au sein de celui-ci.

Pour cela, il faut suivre les étapes suivantes :

- Renseigner des champs personnalisés dans **Squash TM** et les associer au projet portant le plan de tests à exécuter.
- S'assurer que les champs *code* des paramètres correspondent aux noms des variables d'environnement existant dans le script *Cypress*.

**Note :** **Squash TM** ajoute un préfixe au *code* du champ personnalisé transmis. Assurez-vous de le prendre en compte. Voir la [documentation](#) de **Squash TM** pour plus d'information.

Ci-dessous un exemple d'un fichier de test *Cypress* et l'automatisation du cas de test **Squash TM** associé :

The image displays two screenshots. The top screenshot shows a Cypress test file named `calculator.spec.js` with the following content:

```
1 describe('Calculator', () => {
2   it('add', () => {
3     var a = 5
4     var b = 12
5     expect(a + b).to.equal(parseInt(Cypress.env('CP6_CUF_add_result')))
6   })
7   it('mult', () => {
8     var a = 2
9     var b = 4
10    expect(a * b).to.equal(parseInt(Cypress.env('IT_CUF_mult_result')))
11  })
12  it('sub', () => {
13    var a = 10
14    var b = 5
15    expect(a - b).to.equal(parseInt(Cypress.env('TC_CUF_sub_result')))
16  })
17  it('div', () => {
18    var a = 10
19    var b = 5
20    expect(a / b).to.equal(parseInt(Cypress.env('TS_CUF_div_result')))
21  })
22 })
```

The bottom screenshot shows the Squash Test Case interface for 'test\_case\_1'. It includes a sidebar with a tree view showing 'Test Project-1' > 'Cypress' > 'test\_case\_1'. The main panel displays the test case details:

- Informations:** Created on 08/03/2021 10:14 (admin), Modified on 08/03/2021 10:15 (admin). Buttons: Renommer, Imprimer.
- Description (ID = 242):** Format: Classique, Référence: (Cliquer pour éditer...), Description: (Cliquer pour éditer...), Statut: 1-En cours de rédaction.
- Attributs:** sub\_result: 5.
- Automatisation:** Technologie du test automatisé: Cypress, URL du dépôt de code source: https://my-scm/myrepo/cypressParamCalculator (master), Référence du test automatisé: cypressParamCalculator/cypress/integration/calculator.spec.js.
- Champs personnalisés:** add\_result: 18, mult\_result: 8, div\_result: 2.

## Automatisation avec JUnit

### Référence du test

Pour lier un cas de test **Squash TM** à un test automatisé *JUnit*, le champ *Référence* du test automatisé du bloc *Automatisation* du cas de test doit avoir la forme suivante :

[1] / [2] # [3]

Avec :

- [1] : Nom du projet sur le dépôt de source.
- [2] : Nom qualifié de la classe de test.

— [3] : Nom de la méthode à tester dans la classe de test.

Ci-dessous un exemple de classe de test *JUnit* et l'automatisation du cas de test **Squash TM** associé :

```

1  package squash.tfauto;
2
3  import org.junit.jupiter.api.Assertions;
4  import org.junit.jupiter.api.DisplayName;
5  import org.junit.jupiter.api.Test;
6
7  @DisplayName("Calculator")
8  public class CalculatorTest {
9
10     .....
11
12     .....
13
14
15     .....
16
17     .....
18
19     @Test
20     public void multFailure(){
21         int first = 2;
22         int second = 4;
23         Assertions.assertTrue((first*second)==6, "Le résultat du calcul est incorrect. " + first + " *
24     }

```

The screenshot displays the Squash TM web interface. On the left, a sidebar titled "Espace Cas de test" shows a tree view of test projects, with "Test Project-1" expanded to show "test\_case\_1". The main area is titled "test\_case\_1" and contains a form for configuring the test case. The form includes fields for "Créé le" (09/03/2021 05:05 (admin)) and "Modifié le" (09/03/2021 05:06 (admin)). Below these are tabs for "Informations", "Pas de test", "Paramètres", "Pièces jointes", and "Exécutions". The "Informations" tab is active, showing a "Description [ID = 249]" section with "Format : Classique" and "Référence : (Cliquez pour éditer...)". The "Automatisation" section below shows "Technologie du test automatisé : JUnit", "URL du dépôt de code source : https://my-scm/myrepo/junitCalc (master)", and "Référence du test automatisé : junitCalc/squash.tfauto.CalculatorTest#multFailure".

## Automatisation avec Robot Framework

### 1. Référence du test

Pour lier un cas de test **Squash TM** à un test automatisé, le champ *Référence* du test automatisé du bloc *Automatisation* du cas de test doit avoir une forme spécifique au framework de test utilisé :

[1] / [2] # [3]















Avec :

- [1] : Nom du projet sur le dépôt de source.
- [2] : Chemin et nom du fichier de test *Robot Framework* à partir de la racine du projet (avec son extension *.robot*).
- [3] : Nom du cas de test à exécuter dans le fichier *.robot*.

### 2. Nature des paramètres Squash TM exploitables

Les paramètres **Squash TM** exploitables dans un script *Robot Framework* vont différer suivant si vous utilisez les composants **Squash DEVOPS Community** ou **Squash DEVOPS Premium**.

Voici le tableau des paramètres exploitables :

Nature	Clé	Community	Premium
Nom du jeu de donnée	DSNAME		
Paramètre d'un jeu de donnée	DS_[nom]		
Référence du cas de test	TC_REF		
CUF cas de test	TC_CUF_[code]		
CUF itération	IT_CUF_[code]		
CUF campagne	CPG_CUF_[code]		
CUF suite de tests	TS_CUF_[code]		

*Légende :*

- CUF : *Custom Field / Champ personnalisé*
- [code] : *Valeur renseignée dans le champ "Code" d'un CUF*
- [nom] : *nom du paramètre tel que renseigné dans Squash TM*



### 3. Utilisation de paramètres Squash TM

Il est possible lors de l'exécution d'un cas de test **Squash TM** automatisé avec *Robot Framework* d'exploiter des paramètres **Squash TM** au sein de celui-ci.

Pour cela, il faut suivre les étapes suivantes :

- Renseigner des champs personnalisés dans **Squash TM** et les associer au projet portant le plan de tests à exécuter.
- Installer sur le/les environnement(s) d'exécution *Robot Framework* la librairie python *squash-tf-services*. Elle est accessible par le gestionnaire de package *pip* et peut s'installer en exécutant la ligne de commande suivante :

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

- Importer la librairie au sein du fichier `.robot` dans la section *Settings* :

```
Library squash_tf.TFParamService
```

- Vous pouvez ensuite récupérer la valeur d'un paramètre **Squash TM** en faisant appel au mot-clef suivant :

```
Get Param <clé du paramètre>
```

Ci-dessous un exemple de fichier de test *Robot Framework* et l'automatisation du cas de test **Squash TM** associé :

The image displays two parts of the workflow. The top part is a code editor showing a `robot-param-demo / parmTest.robot` file. The code includes settings for documentation, libraries (`squash_tf.TFParamService` and `conditionalHang.py`), and a test case named 'Parameter Test'. The test case uses the `Get Param` keyword to retrieve the value of `TC_REFERENCE` and then performs a conditional hang based on that value.

The bottom part is a screenshot of the Squash TM web interface. It shows the 'Espace Cas de test' (Test Case Space) for a project named '42 - test\_case\_robot\_framework'. The interface includes a sidebar with a tree view showing the test case structure. The main panel displays details for the selected test case, including its description, format (Classique), reference (42), and automation configuration. The automation configuration specifies the use of Robot Framework, the source code repository URL, and the specific test case reference.

## Automatisation avec SoapUI

### Référence du test

Pour lier un cas de test **Squash TM** à un test automatisé *SoapUI*, le champ *Référence* du test automatisé du bloc *Automatisation* du cas de test doit avoir la forme suivante :

[1] / [2] # [3] # [4]

Avec :

- [1] : Nom du projet sur le dépôt de source.
- [2] : Chemin et nom du fichier de test *SoapUI* à partir de la racine du projet (avec son extension *.xml*).
- [3] : Nom de la Suite de test *SoapUI* contenant le cas de test.
- [4] : Nom du cas de test à exécuter.

Ci-dessous un exemple de fichier de test *SoapUI* et l'automatisation du cas de test **Squash TM** associé :

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <soapui-project id="9b9b79db-37b7-471a-a803-d99ba63c95d7" activeEnvironment="Default" openWeatherTest="resourceRoot"" soapui-version="5.6.0" xmlns:con="http://www.soapui.org/config" ?>
3   <!-->
4   <!-->
5   <!-->
6   <!-->
7   <!-->
8   <!-->
9   <!-->
10  <!-->
11  <!-->
12  <!-->
13  <!-->
14  <!-->
15  <!-->
16  <!-->
17  <!-->
18  <!-->
19  <!-->
20  <!-->
21  <!-->
22  <!-->
23  <!-->
24  <!-->
25  <!-->
26  <!-->
27  <!-->
28  <!-->
29  <!-->
30  <!-->
31  <!-->
32  <!-->
33  <!-->
34  <!-->
35  <!-->
36  <!-->
37  <!-->
38  <!-->
39  <!-->
40  <!-->
41  <!-->
42  <!-->
43  <!-->
44  <!-->
45  <!-->
46  <!-->
47  <!-->
48  <!-->
49  <!-->
50  <!-->
51  <!-->
52  <!-->
53  <!-->
54  <!-->
55  <!-->
56  <!-->
57  <!-->
58  <!-->
59  <!-->
60  <!-->
61  <!-->
62  <!-->
63  <!-->
64  <!-->
65  <!-->
66  <!-->
67  <!-->
68  <!-->
69  <!-->
70  <!-->
71  <!-->
72  <!-->
73  <!-->
74  <!-->
75  <!-->
76  <!-->
77  <!-->
78  <!-->
79  <!-->
80  <!-->
81  <!-->
82  <!-->
83  <!-->
84  <!-->
85  <!-->
86  <!-->
87  <!-->
88  <!-->
89  <!-->
90  <!-->
91  <!-->
92  <!-->
93  <!-->
94  <!-->
95  <!-->
96  <!-->
97  <!-->
98  <!-->
99  <!-->
100 </soapui-project>
```

### 1.3.2 Déclenchement d'un plan d'exécution depuis Squash TM

#### Déclaration du serveur Squash Orchestrator

Afin de lancer un plan d'exécution manuellement depuis **Squash TM**, il est nécessaire de déclarer le serveur **Squash Orchestrator** qui exécutera les tests automatisés dans les environnements adaptés. Cette déclaration se fait dans la section *Serveurs d'exécution automatisée* de l'espace *Administration* :

**Ajouter un serveur d'exécution automatisée** ✕

Nom :

Type :

Url :

Description : 

**B** *I* U ~~S~~

Police

Taille

ABC

- Nom : Le nom du serveur tel qu'il apparaîtra dans l'espace *Cas de test*.
- Type : Sélectionnez *squashAutom* dans la liste déroulante.
- Url : L'adresse du service *Receptionist* de **Squash Orchestrator**.

**Avertissement :** Le service *bus d'évènements* de **Squash Orchestrator** doit impérativement être accessible à la même url que le service *Receptionist* mais sur le port 38368.

Une fois le serveur créé, vous pouvez préciser un token servant pour l'authentification au serveur.

Protocole d'authentification

token authentication ▾

Politique d'authentification

Identifiants

Jeton

Enregistrer

**Note :** La présence d'un token est obligatoire pour la bonne exécution de tests automatisés depuis **Squash TM**. Dans le cas où l'authentification au serveur d'automatisation n'est pas requise par ce dernier, il faut quand même renseigner une valeur quelconque de token dans **Squash TM**

## Exécution de la suite automatisée

L'exécution d'un plan d'exécution automatisée se déroule de la façon habituelle dans **Squash TM** :

- Accédez au plan d'exécution de l'itération ou de la suite de test choisie.
- Exécutez les tests automatisés en utilisant un des boutons de l'image ci-dessous :

1 - iter

Créé le : 24/02/2021 16:34 (admin)  
Modifié le : 12/03/2021 15:03 (admin)

Lancer Lancer les tests automatisés Suites de tests Renommer

Tableau de bord Informations Plan d'exécution Suites automatisées

Filtrer Réordonner Suites de tests Statut Assigner Ajouter Retirer du plan d'exécution

#	Emplacement	Mode	Ref.	Test	Imp.	Suite de tests	Statut	% succès	Utilisateur	Dernière exécution
1	projetSquashAutom			testAutom	F	-	à exécuter	0 %	-	-

Nouvelle exécution Exécuter automatiquement

Afficher 50 éléments :

Bouton [Exécuter automatiquement]

Bouton [Lancer les tests automatisés]

Bouton [Lancer une exécution]

Cliquer pour afficher les options :  
Tous et Sélection

- Une popup de suivi d'exécution apparaît.

**Note :** La popup de suivi contient une nouvelle section qui rappelle le nom des tests en cours d'exécution par **Squash Orchestrator**. Cependant il n'y pas de suivi d'avancement dans la version actuelle.

### 1.3.3 Remontées de résultats après exécution d'un plan d'exécution Squash TM

Quel que soit la façon dont le plan d'exécution est déclenché (depuis **Squash TM** ou depuis un pipeline *Jenkins*), vous avez en fin d'exécution une remontée de résultats au sein de **Squash TM** qui diffère suivant si vous êtes sous licence **Squash AUTOM Community** ou **Squash AUTOM Premium**.

#### Squash AUTOM Community

Après exécution d'un plan d'exécution **Squash TM** (itération ou suite de tests), les informations suivantes sont mises à jour :

- Mise à jour du statut des ITPI.
- Mise à jour du statut de la suite automatisée.
- Le rapport au format *Allure* correspondant à l'ensemble des tests exécutés.
- Les rapports d'exécution des différents ITPI sont accessibles depuis l'onglet de consultation des suites automatisées :

The screenshot shows the Squash TM interface. On the left is a sidebar with a tree view showing 'Test Project-1' and its sub-items. The main panel has a top bar with navigation links and a table of test runs. An 'Info' dialog box is open, displaying a list of report files generated during the execution.

#	Créé le	Statut		Créé par	Modifié le
1	09/03/2021 15:37	échec	Voir le détails des exécutions	tfserver	09/03/2021 15:38

Liste des rapports d'exécution:

- allure-report.tar
- test\_case\_cucumber[285]-html-report.tar
- test\_case\_cucumber[285]-report.json
- test\_case\_cucumber[285]-report.xml
- test\_case\_cucumber[286]-html-report.tar
- test\_case\_cucumber[286]-report.json
- test\_case\_cucumber[286]-report.xml
- test\_case\_cucumber[287]-html-report.tar
- test\_case\_cucumber[287]-report.json
- test\_case\_cucumber[287]-report.xml
- test\_case\_cypress[288]-calculator-report.xml
- test\_case\_junit[289]-TEST-squash.tfauto.CalculatorTest.xml
- test\_case\_junit[289]-squash.tfauto.CalculatorTest.txt
- test\_case\_robot\_framework[290]-log.html
- test\_case\_robot\_framework[290]-output.xml
- test\_case\_robot\_framework[290]-report.html
- test\_case\_soapui[291]-TEST-ForecastSuite.xml

**Note :** Tous les résultats de la suite automatisée sont compilés dans un rapport au format *Allure*, disponible dans la liste de rapports sous forme d'archive *.tar*.

Cependant dans cette version *1.0.0-alpha2*, les rapports de test *Robot Framework* ne peuvent pas être inclus dans ce rapport. Si la suite automatisée ne contient que des tests *Robot Framework*, une archive sera néanmoins générée mais le rapport sera vide.

Pour plus d'information sur la façon d'exploiter ce rapport et les possibilités de personnalisation, veuillez consulter la [documentation Allure](#).

---

Cependant, voici ce qu'il ne se passe pas :

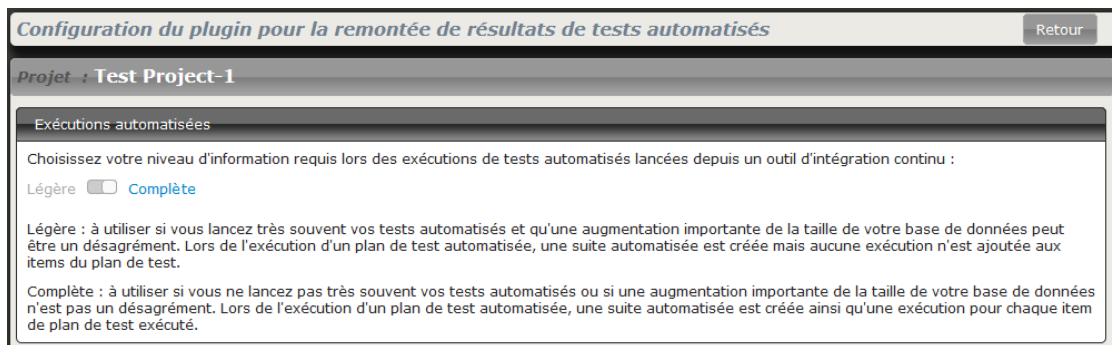
- Création d'une nouvelle exécution pour chaque ITPI qui a été exécuté.
- 

### Squash AUTOM Premium

Si vous disposez des composants **Squash AUTOM Premium**, vous avez accès à deux types de remontées d'informations :

- Légère (valeur par défaut).
- Complète.

Le choix du type de remontée se fait par projet en accédant à la configuration du plugin **Squash TM Result Publisher** depuis l'onglet *Plugins* de la page de consultation d'un projet :



### Remontée d'information Légère

En choisissant la remontée d'information *Légère*, les informations suivantes sont mises à jour après exécution d'un plan d'exécution **Squash TM** (itération ou suite de tests) :

- Mise à jour du statut des ITPI.
- Mise à jour du statut de la suite automatisée.
- Le rapport au format *Allure* correspondant à l'ensemble des tests exécutés.
- Les rapports d'exécution des différents ITPI sont accessibles depuis l'onglet de consultation des suites automatisées :

The screenshot shows the 'Espace Campagnes' interface. On the left, a sidebar shows a tree view with 'Test Project-1' containing 'Folder Test 1' and 'Campaign Test 1', which in turn contains '1 - Iteration'. The main area is titled '1 - Iteration' and shows a table of test results. The first row indicates a failure ('échec') on 09/03/2021 at 15:37. An 'Info' modal window is open, displaying a list of generated report files in Allure format.

**Info**

Liste des rapports d'exécution:

- allure-report.tar
- test\_case\_cucumber[285]-html-report.tar
- test\_case\_cucumber[285]-report.json
- test\_case\_cucumber[285]-report.xml
- test\_case\_cucumber[286]-html-report.tar
- test\_case\_cucumber[286]-report.json
- test\_case\_cucumber[286]-report.xml
- test\_case\_cucumber[287]-html-report.tar
- test\_case\_cucumber[287]-report.json
- test\_case\_cucumber[287]-report.xml
- test\_case\_cypress[288]-calculator-report.xml
- test\_case\_junit[289]-TEST-squash.tfauto.CalculatorTest.xml
- test\_case\_junit[289]-squash.tfauto.CalculatorTest.txt
- test\_case\_robot\_framework[290]-log.html
- test\_case\_robot\_framework[290]-output.xml
- test\_case\_robot\_framework[290]-report.html
- test\_case\_soapui[291]-TEST-ForecastSuite.xml

Fermer

**Note :** Tous les résultats de la suite automatisée sont compilés dans un rapport au format *Allure*, disponible dans la liste de rapports sous forme d'archive *.tar*.

Cependant dans cette version *1.0.0.alpha2*, les rapports de test *Robot Framework* ne peuvent pas être inclus dans ce rapport. Si la suite automatisée ne contient que des tests *Robot Framework*, une archive sera néanmoins générée mais le rapport sera vide.

Pour plus d'information sur la façon d'exploiter ce rapport et les possibilités de personnalisation, veuillez consulter la [documentation Allure](#).

Cependant, voici ce qu'il ne se passe pas :

- Création d'une nouvelle exécution pour chaque ITPI qui a été exécuté.

## Remontée d'information Complète

En choisissant la remontée d'information *Complète*, les informations suivantes sont mises à jour après exécution d'un plan d'exécution **Squash TM** (itération ou suite de tests) :

- Mise à jour du statut des ITPI.
- Création d'une nouvelle exécution pour chaque ITPI.
- Mise à jour du statut de la suite automatisée.
- Le rapport au format *Allure* correspondant à l'ensemble des tests exécutés.
- Les rapports d'exécution des différentes exécutions sont accessibles depuis l'onglet de consultation des suites automatisées ou depuis l'écran de consultation de l'exécution (ils sont présents dans les pièces jointes).

**Espace Campagnes**

Créé le : 09/03/2021 16:35 (admin)  
Modifié le : 09/03/2021 16:35 (admin)

Relancer Suites de tests Renommer

Tableau de bord Informations Plan d'exécution Suites automatisées Pièces jointes

Filtrer Réordonner Suites de tests Statut Assigner Ajouter Retirer du plan d'exécution

#	Emplacement	Ref.	Test	Imp.	Jeux de données	Suite de tests	Statut	% succès	Utilisateur	Dernière exécution
1	Test Project-1	:	test_case_cucumber	F	tag1	-	succès	100 %	tfserver (tfserver)	09/03/2021 16:40
<a href="#">Exec. 1 : test_case_cucumber</a> tag1 succès tfserver 09/03/2021 15:40 <a href="#">Nouvelle exécution</a>										
2	Test Project-1	:	test_case_cucumber	F	tag2	-	succès	100 %	tfserver (tfserver)	09/03/2021 16:40
<a href="#">Exec. 1 : test_case_cucumber</a> tag2 succès tfserver 09/03/2021 15:40 <a href="#">Nouvelle exécution</a>										
3	Test Project-1	:	test_case_cucumber	F	tag3	-	échec	0 %	tfserver (tfserver)	09/03/2021 16:40



**Espace Campagnes** □ Filtre global Administration Mon compte (.admin.) Déconnexion

**Exécution : #1 - test\_case\_cucumber** Retour

Description :

Statut : ● 1-En cours de rédaction

Jeu de données : tag1

Script auto : (supprimé)

---

**Attributs**

Importance : ▼ 4-Faible

Nature : Non définie

Type : Non défini

sub\_result (issu du cas de test) : 5

---

**Prérequis**

---

**Exigences vérifiées**

#	Projet	ID version	Référence	Exigence	Criticité
Aucun élément à afficher					

Affichage 0 à 0 sur 0 élément(s)

---

**Champs personnalisés**

---

**Résumé du résultat**

---

**Scénario d'exécution**

#	Action	Résultat att.	Statut	Dernière exec.	Utilisateur	Commentaires	PJ.	Exec.
Aucun élément à afficher								

Afficher 50 éléments :


---


**Commentaires**


(Cliquez pour éditer...)

---

**Pièces jointes** Ajouter une pièce jointe Organiser


[test\\_case\\_cucumber\[285\]-report.xml](#)


[test\\_case\\_cucumber\[285\]-report.json](#)


[test\\_case\\_cucumber\[285\]-html-report.tar](#)

**Note :** Tous les résultats de la suite automatisée sont compilés dans un rapport au format *Allure*, disponible dans la liste de rapports sous forme d'archive *.tar*.

Cependant dans cette version *1.0.0.alpha2*, les rapports de test *Robot Framework* ne peuvent pas être inclus dans ce rapport. Si la suite automatisée ne contient que des tests *Robot Framework*, une archive sera néanmoins générée mais le rapport sera vide.

Pour plus d'information sur la façon d'exploiter ce rapport et les possibilités de personnalisation, veuillez consulter la [documentation Allure](#).

Ce guide vous présente les possibilités offerte par la version *1.0.0.alpha2* de **Squash AUTOM**.

**Avertissement :** Cette version est destinée à des POC et est donc utilisable dans un contexte hors production (notamment avec un **Squash TM** dont la base de données est neuve ou la réplication d'une base existante).

Cette version *1.0.0.alpha2* met à votre disposition deux composants :

- **Squash Orchestrator** : il s'agit d'un outil composé d'un ensemble de micro-services exploitables via l'envoi d'un plan d'exécution sous un formalisme bien précis, le PEAC (Plan d'Exécution «as code»), afin d'orchestrer des exécutions de tests automatisés.
- **Plugin Result Publisher pour Squash TM** : ce plugin pour **Squash TM** permet la remontée d'informations vers **Squash TM** en fin d'exécution d'un plan d'exécution **Squash TM** par l'orchestrateur Squash.
- **Plugin Squash AUTOM pour Squash TM** : ce plugin pour **Squash TM** permet d'exécuter des tests automatisés depuis ce dernier via **Squash Orchestrator**.

## 2.1 Guide d'Installation

- *Squash Orchestrator*
- *Plugin Test Plan Retriever pour Squash TM*
- *Plugin Squash DEVOPS pour Jenkins*

### 2.1.1 Squash Orchestrator

Ce micro-service existe en version **Squash DEVOPS Community** et **Squash DEVOPS Premium**. Ils sont inclus dans l'image *Docker* de **Squash Orchestrator**. Pour des détails sur le déploiement de **Squash Orchestrator** et l'activation du micro-service **Squash TM Generator** en version **Community** ou **Premium**, merci de consulter la documentation de **Squash Orchestrator** (*Squash Orchestrator Documentation – 1.0.0.alpha2* version .pdf) téléchargeable depuis <https://www.squashtest.com/community-download>.

## 2.1.2 Plugin Test Plan Retriever pour Squash TM

Le plugin existe en version **Community** (*squash.tm.rest.test.plan.retriever.community-1.0.0.alpha2.jar*) librement téléchargeable ou **Premium** (*squash.tm.rest.test.plan.retriever.premium-1.0.0.alpha2.jar*) accessible sur demande.

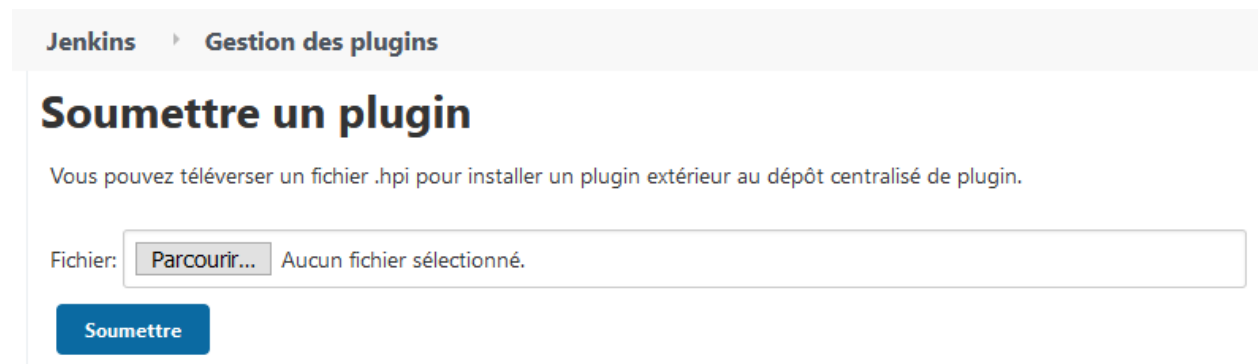
Pour l'installation, merci de vous reporter au protocole d'installation d'un plugin **Squash TM** (<https://sites.google.com/a/henix.fr/wiki-squash-tm/installation-and-exploitation-guide/2—installation-of-squash-tm/7—jira-plug-in>).

**Avertissement :** Ce plugin est compatible avec une version *1.22.2.RELEASE* de **Squash TM**.

## 2.1.3 Plugin Squash DEVOPS pour Jenkins

Le plugin est sous la forme d'un fichier **.hpi** (*squash-devops-1.0.0.alpha2.hpi*) librement téléchargeable depuis <https://www.squashtest.com/community-download>.

Pour l'installation, soumettez le plugin depuis l'onglet *Avancé* de l'écran de gestion des plugins de *Jenkins* :



Jenkins > Gestion des plugins

### Soumettre un plugin

Vous pouvez téléverser un fichier .hpi pour installer un plugin extérieur au dépôt centralisé de plugin.

Fichier:  Aucun fichier sélectionné.

**Avertissement :** Ce plugin est compatible avec une version *2.164.1* ou supérieure de *Jenkins*.

## 2.2 Appel au Squash Orchestrator depuis un pipeline Jenkins

- Configuration d'un Squash orchestrator dans Jenkins
- Appel au Squash Orchestrator depuis un pipeline Jenkins

## 2.2.1 Configuration d'un Squash orchestrator dans Jenkins

Pour accéder à l'espace de configuration du **Squash Orchestrator**, il faut tout d'abord se rendre dans l'espace *Configurer le système* du *System Configuration*, accessible par l'onglet *Administrer Jenkins* :

### System Configuration



#### Configurer le système

Configurer les paramètres généraux et les chemins de fichiers.



#### Configuration globale des outils

Configurer les outils, leur localisation et les installeurs automatiques.



#### Gestion des plugins

Ajouter, supprimer, activer ou désactiver des plugins qui peuvent étendre les fonctionnalités de Jenkins.

mises à jour disponibles



#### Configuration files

administration des fichiers de configurations tels que settings.xml pour Apache Maven.

Un panel nommé *Squash Orchestrator servers* sera ensuite disponible :

### Squash Orchestrator servers

Server id	<input type="text" value="46705135"/>
Server name	<input type="text" value="defaultServer"/>
Receptionist endpoint URL	<input type="text" value="http://127.0.0.1:7774"/>
Workflow Status endpoint URL	<input type="text" value="http://127.0.0.1:7775"/>
Credential	<input type="text" value="- none -"/> <input type="button" value="Add"/>
Workflow Status poll interval	<input type="text" value="2S"/>
Workflow creation timeout	<input type="text" value="5S"/>

- **Server id** : Cet ID est généré automatiquement et ne peut être modifié. Il n'est pas utilisé par l'utilisateur.
- **Server name** : Ce nom est défini par l'utilisateur. C'est celui qui sera mentionné dans le script du pipeline lors d'une exécution de workflow.
- **Receptionist endpoint URL** : L'adresse du micro-service *receptionist* de l'orchestrateur, avec son port tel que défini au lancement de l'orchestrateur. Reportez-vous à la documentation de **Squash Orchestrator** pour plus de détails.
- **Workflow Status endpoint URL** : L'adresse du micro-service *observer* de l'orchestrateur, avec son port tel que défini au lancement de l'orchestrateur. Reportez-vous à la documentation de **Squash Orchestrator** pour plus de détails.
- **Credential** : Credential *Jenkins* de type *Secret text* contenant un *JWT Token* permettant de s'authentifier auprès de l'orchestrateur. Reportez-vous à la documentation de **Squash Orchestrator** pour plus de détails sur l'accès sécurisé à l'orchestrateur.
- **Workflow Status poll interval** : Ce paramètre correspond au temps de mise à jour du statut du workflow.
- **Workflow creation timeout** : Timeout sur la réception du PEAC par le *receptionist* côté orchestrateur.

## 2.2.2 Appel au Squash Orchestrator depuis un pipeline Jenkins

Une fois qu'il y a au moins un **Squash Orchestrator** configuré dans *Jenkins*, il est possible de faire appel à l'orchestrateur depuis un job *Jenkins* de type pipeline grâce à une méthode de pipeline dédiée.

Ci-dessous, un exemple de pipeline simple utilisant la méthode d'appel à l'orchestrateur :

```
node {
    stage 'Stage 1 : sanity check'
    echo 'OK pipelines work in the test instance'
    stage 'Stage 2 : steps check'
    configFileProvider([configFile(
fileId: '600492a8-8312-44dc-ac18-b5d6d30857b4',
targetLocation: 'testWorkflow.json'
)]) {
        def workflow_id = runSquashWorkflow(
            workflowPathName: 'testWorkflow.json',
            workflowTimeout: '20S',
            serverName: 'defaultServer'
        )
        echo "We just ran The Squash Orchestrator workflow $workflow_id"
    }
}
```

La méthode *runSquashWorkflow* permet de transmettre un PEAC à l'orchestrateur pour exécution.

Elle dispose de 3 paramètres :

- **workflowPathName** : Le chemin vers le fichier contenant le PEAC. Dans le cas présent, le fichier est injecté via le plugin *Config File Provider*, mais il est également possible de l'obtenir par d'autres moyens (récupération depuis un SCM, génération à la volée dans un fichier, ...).
- **workflowTimeout** : Timeout sur l'exécution des actions. Ce paramètre intervient par exemple si un environnement n'est pas joignable (ou n'existe pas), ou si une action n'est pas trouvée par un *actionProvider*. Il est à adapter en fonction de la durée d'exécution attendue des différents tests du PEAC.
- **serverName** : Nom du serveur **Squash Orchestrator** à utiliser. Ce nom est celui défini dans l'espace de configuration *Squash Orchestrator servers* de *Jenkins*.

## 2.3 Récupération d'un plan d'exécution Squash TM depuis un PEAC

- *Prérequis*
- *Intégration de l'étape de récupération d'un plan d'exécution Squash TM dans un PEAC*
- *Paramètres Squash TM exploitables dans un test automatisé*
- *Remontée d'informations vers Squash TM en fin d'exécution*

**Squash DEVOPS** vous donne la possibilité de récupérer un plan d'exécution de tests automatisés définis dans **Squash TM** depuis un PEAC. Ce PEAC pouvant être déclenché depuis un pipeline *Jenkins* par exemple (voir la [page correspondante](#) de ce guide).

### 2.3.1 Prérequis

Afin de pouvoir récupérer un plan d'exécution **Squash TM** depuis un PEAC, vous avez besoin d'avoir effectué les actions suivantes dans **Squash TM** :

- Création d'un utilisateur appartenant au groupe *Serveur d'automatisation de tests*.
- Création d'un plan d'exécution (itération ou suite de tests) contenant au moins un ITPI lié à un cas de test automatisé suivant les instructions du guide utilisateur **Squash AUTOM** (voir [ici](#))

### 2.3.2 Intégration de l'étape de récupération d'un plan d'exécution Squash TM dans un PEAC

Pour récupérer un plan d'exécution **Squash TM** dans un PEAC, il faut faire appel à l'action *generator* correspondante.

Voici ci-dessous un exemple simple de PEAC au format *Json* permettant de récupérer un plan d'exécution **Squash TM** :

```
{
  "apiVersion": "opentestfactory.org/v1alpha1",
  "kind": "Workflow",
  "metadata": {
    "name": "Simple Workflow"
  },
  "defaults": {
    "runs-on": "ssh"
  },
  "jobs": {
    "explicitJob": {
      "runs-on": "ssh",
      "generator": "tm.squashtest.org/tm.generator@v1",
      "with": {
        "testPlanUuid": "1e2ae123-6b67-44b2-b229-274ea17ad489",
        "testPlanType": "Iteration",
        "squashTMUrl": "https://mySquashTMInstance.org/squash",
        "squashTMAutomatedServerLogin": "tfserver",
        "squashTMAutomatedServerPassword": "tfserver"
      }
    }
  }
}
```

Un step *generator* **Squash TM** doit contenir les paramètres suivants :

- `testPlanType` : Correspond au type du plan de test à récupérer dans **Squash TM**. Seules les valeurs *Iteration* et *TestSuite* sont acceptées.
- `testPlanUuid` : Correspond à l'UUID du plan de test souhaité. Celui-ci peut être récupéré dans le panneau Information de l'itération ou de la suite de tests souhaitée dans **Squash TM**.
- `squashTMUrl` : URL du **Squash TM** à viser.
- `squashTMAutomatedServerLogin` : Nom de l'utilisateur du groupe *Serveur d'automatisation de tests* à utiliser dans **Squash TM**.
- `squashTMAutomatedServerPassword` : Mot de passe de l'utilisateur du groupe *Serveur d'automatisation de tests* à utiliser dans **Squash TM**.

[Champs Optionnels] :

- `tagLabel` : Spécifique à la version **Premium** - Correspond au nom du champ personnalisée de type *tag* sur lequel on souhaite filtrer les cas de test à récupérer. Il n'est pas possible d'en spécifier plusieurs.
- `tagValue` : Spécifique à la version **Premium** - Correspond à la valeur du champ personnalisée de type *tag* sur lequel on souhaite filtrer les cas de test à récupérer. Il est possible d'indiquer plusieurs valeurs séparées par des "||" (Exemple : valeur1|valeur2). Il faut au moins l'une des valeurs pour que le cas de test soit pris en compte.

**Avertissement :** Si l'un des deux champs `tagLabel` ou `tagValue` est présent, alors l'autre champ **doit** également être renseigné.

### 2.3.3 Paramètres Squash TM exploitables dans un test automatisé

En exécutant un PEAC avec récupération d'un plan d'exécution **Squash TM**, ce dernier transmet différentes informations sur les ITPI qu'il est possible d'exploiter dans un cas de test *Cucumber*, *Cypress*, ou *Robot Framework*.

Pour plus d'information veuillez consulter la section *Exploitation de paramètres Squash TM* de la documentation de **Squash AUTOM**, ainsi que la section dédiée à l'automatisation du framework de test souhaité.

### 2.3.4 Remontée d'informations vers Squash TM en fin d'exécution

La nature de la remontée d'informations à **Squash TM** en fin d'exécution d'un plan d'exécutions **Squash TM** va dépendre de si vous êtes sous licence **Squash AUTOM Community** ou **Squash AUTOM Premium**.

Consultez le guide utilisateur **Squash AUTOM** pour plus d'informations (voir *ici*).

Ce guide vous présente les possibilités offerte par la version *1.0.0.alpha2* de **Squash DEVOPS**.



<p><b>Avertissement :</b> Cette version est destinée à des POC et est donc utilisable dans un contexte hors production (notamment avec un <b>Squash TM</b> dont la base de données est neuve ou la réplication d'une base existante).</p>
---

Cette version *1.0.0.alpha2* met à votre disposition les composants suivants :

- **Micro-service Squash TM Generator pour Squash Orchestrator** : il s'agit d'un micro-service de **Squash Orchestrator** permettant la récupération d'un plan d'exécution **Squash TM** au sein d'un PEAC (Plan d'Exécution «as code»). Consultez le guide utilisateur de *Squash AUTOM* pour plus d'informations sur **Squash Orchestrator** et les PEAC.
- **Plugin Test Plan Retriever pour Squash TM** : ce plugin pour **Squash TM** permet l'envoi à **Squash Orchestrator** de détails sur un plan d'exécution **Squash TM**.
- **Plugin Squash DEVOPS pour Jenkins** : ce plugin pour *Jenkins* facilite l'envoi d'un PEAC à Squash Orchestrator depuis un pipeline *Jenkins*.