Squash Autom & Squash Devops

Version 1.0.0-alpha1

squashtest

nov. 30, 2021

Table des matières

Avertissement : Ce site est obsolète, la documentation de Squash AUTOM et Squash DEVOPS est désormais https://autom-devops-fr.doc.squashtest.com/.

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.

Avertissement : Ce site est obsolète, la documentation de Squash AUTOM et Squash DEVOPS est désormais https://autom-devops-fr.doc.squashtest.com/.

CHAPITRE 1

Squash AUTOM

1.1 Guide d'Installation

1.1.1 Squash Orchestrator

Présentation

Squash Orchestrator vous permet de diriger et coordonner les différents composants de la chaîne d'exécution de vos tests autoamtisés (environnements d'exécution, automates, reporting…) Il est basé sur l'OpenTestFactory tout en ajoutant un ensemble de micro-services pour étendre ses possibilités, notamment le fait d'exploiter des plans d'exécutions Squash TM ou d'effectuer du reporting vers ce dernier.

Installation

Squash Orchestrator se présente sous la forme d'un ensemble de services fonctionnant ensemble. Ils peuvent ou non être lancés sur une même machine, et être ou non démarrés simultanément.

Le seul prérequis est que le service *EventBus*, qui permet la communication entre les différents micro-services, soit disponible au moment du lancement des autres micro-services.

Pour faciliter l'installation de **Squash Orchestrator**, une image docker « all-in-one » est mise à disposition. Elle contient l'ensemble des services nécessaires de l'OpenTestFactory et une partie des services spécifiques Squash.

Pour récupérer la dernière version de l'image de **Squash Orchestrator**, la commande suivante est à exécuter :

```
docker pull squashtest/squash-orchestrator:latest
```

Usage

Configuration de l'image

La commande suivante permet de démarrer **Squash Orchestrator** en lui permettant d'utiliser un environnement d'exécution existant, avec auto-génération des clés de trusted keys (ce qui n'est **pas** recommandé dans un environnement de production).

```
docker run -d \
         --name orchestrator \
         -p 7774:7774 \
         -p 7775:7775 \
         -p 7776:7776 \
         -p 38368:38368 \
         -e SSH_CHANNEL_HOST=the_environment_ip_or_hostname \
         -e SSH_CHANNEL_USER=user \
         -e SSH_CHANNEL_PASSWORD=secret \
         -e SSH_CHANNEL_TAGS=ssh,linux,robotframework \
          squashtest/squash-orchestrator:latest
```

Cette commande expose les services suivants sur les ports correspondants :

- *receptionnist* (port 7774)
- *observer* (port 7775)
- *killswitch* (port 7776)
- *eventbus* (port 38368)

Lancement de l'image en mode Premium

Si vous disposez d'une licence **Squash AUTOM Premium** et avez déployé la version **Premium** du plugin Squash TM *Result Publisher*, vous devez démarrez **Squash Orchestrator** en mode **Premium** pour profitez du comportement **Premium** pour la remontée de résultats vers **Squash TM**. Pour cela, il faut ajouter le paramètre suivant dans la commande de démarrage de **Squash Orchestrator** : `-e SQUASH_LICENCE_TYPE=premium`

Pour une configuration plus complète de **Squash Orchestrator**, vous pouvez vous reporter à la documentation de l'OpenTestFectory qui sert de base au **Squash Orchestrator**.

1.1.2 Image Docker des micro-services exclusifs à Squash AUTOM Premium

Présentation

Posséder une licence Squash AUTOM Premium permet d'avoir accès à une image Docker contenant des micro-services pour **Squash Orchestrator** fournissant les fonctionnalités suivantes :

— Gestion de l'exécution de tests Agilitest

Installation

Afin d'installer l'image Docker des services exclusifs à Squash AUTOM Premium, vous devez récupérer l'image compressée auprès du service support Squash puis exécuter la commande suivante :

```
docker load -i squash-autom-premium-1.0.1.tar.gz
```

Usage

Pour exécuter l'image Docker des services exclusifs à Squash AUTOM Premium, la commande suivante est à exécuter :

```
docker run -e BUS_HOST=<IP ou nom DNS du bus> \
           -e BUS_PORT=<port> -e BUS_TOKEN=<token JWT> \
           --volume /chemin/vers/clef/publiques(s):/etc/squashtf/ \
           docker.squashtest.org/squashtest/squash-autom-premium:1.0.1
```

avec :

— *BUS_HOST* (obligatoire) : l'adresse IP ou le nom DNS du service *EventBus* du **Squash Orchestrator** avec qui les micro-services communiquent.

— *BUS_PORT* (obligatoire) : port du service *EventBus* du **Squash Orchestrator** avec qui les micro-services communiquent.

— *BUS_TOKEN* (facultatif) : token JWT accepté par le service *EventBus* du **Squash Orchestrator** avec qui les micro-services communiquent.

— *–volume* : volume à monter vers un ensemble de clés publiques acceptées pour la validité de tokens JWT. Il est à mettre en place si une validation de token JWT est nécessaire pour l'échange de messages entre micro-services.

1.1.3 Agent OpenTestFactory

Présentation

L'agent OpenTestFactory est un process qui tourne sur un environnement d'exécution. Ce process contacte le **Squash Orchestrator** à intervalle régulier, à la recherche d'ordres à exécuter. S'il y a un ordre en attente, l'agent va l'exécuter puis retourner le résultat à l'orchestrateur.

Installation

L'agent OpenTestFactory est une application Python à installer sur un environnement d'exécution de tests automatisés. Il requiert Python 3.8 ou supérieure. Il fonctionne sur un environnement Linux, MacOs ou Windows.

L'agent se présente comme un simple script. Il possède seulement une dépendance, vers la librairie python `requests` (elle sera installée si elle n'est pas déjà presente sur l'environnement d'exécution).

Afin d'installer l'agent, la commande suivante est à exécuter :

```
pip3 install --upgrade opentf-agent
```

Vous pouvez vérifier l'installation en exécutant la commande suivante :

```
opentf-agent --help
```

Usage

Résumé

```
$ opentf-agent --help
usage: opentf-agent [-h] --tags TAGS --host HOST [--port PORT] [--path_prefix PATH_
↪PREFIX] [--token TOKEN] [--encoding ENCODING] [--script_path SCRIPT_PATH]
                    [--workspace_dir WORKSPACE_DIR] [--name NAME] [--polling_delay␣
↪POLLING_DELAY] [--liveness_probe LIVENESS_PROBE] [--retry RETRY] [--debug]

OpenTestFactory Agent

optional arguments:
  -h, --help            show this help message and exit
  --tags TAGS           a comma-separated list of tags (e.g. windows,robotframework)
  --host HOST           target host with protocol (e.g. https://example.local)
  --port PORT           target port (default to 24368)
  --path_prefix PATH_PREFIX
                        target context path (default to no context path)
  --token TOKEN         token
  --encoding ENCODING   encoding on the console side (defaults to utf-8)
  --script_path SCRIPT_PATH
                        where to put temporary files (defaults to current directory)
  --workspace_dir WORKSPACE_DIR
                        where to put workspaces (defaults to current directory)
  --name NAME           agent name (defaults to "test agent")
  --polling_delay POLLING_DELAY
                        polling delay in seconds (default to 5)
  --liveness_probe LIVENESS_PROBE
                        liveness probe in seconds (default to 300 seconds)
  --retry RETRY         how many time to try joining host (default to 5,
                        0 = try forever)
  --debug               whether to log debug informations.
```

Exemple

En considérant qu'un **Squash Orchestrator** est lancé sur `orchestrator.example.com`, avec un jeton stocké dans la variable d'environnement `TOKEN`, la commande suivante enregistre l'environnement d'exécution Windows et recevra les commandes ciblant windows et/ou robotframework :

```
chcp 65001
opentf-agent --tags windows,robotframework --host https://orchestrator.example.com/ --
↪token %TOKEN%
```

L'agent contactera l'orchestrateur toutes les 5 secondes, et exécutera les commandes receptionnées. La commande `chcp` configure la console en Unicode. Il s'agit d'une spécificité Windows. Cette configuration peut-être nécessaire suivant le framework de test disponible sur l'environnement d'exécution.

1.1.4 Plugin Result Publisher pour Squash TM

Présentation

Ce plugin est à utiliser conjointement avec **Squash Orchestrator** pour permettre la remontée de résultats à **Squash TM** à la fin d'une exécution de tests automatisés d'un plan d'exécution **Squash TM**.

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

Installation

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* ou supérieure de **Squash TM**.

1.1.5 Plugin Squash AUTOM pour Squash TM

Présentation

Ce plugin est à utiliser conjointement avec **Squash Orchestrator** pour permettre l'exécution de tests automatisés via **Squash Orchestrator** depuis **Squash TM**.

Le plugin existe en version **Community** (*plugin.testautomation.squashautom.community-1.0.0.RELEASE.jar*) librement téléchargeable ou **Premium** (*plugin.testautomation.squashautom.premium-1.0.0.RELEASE.jar*) accessible sur demande.

Installation

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* ou supérieure de **Squash TM**.

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.RELEASE* version `.pdf`) accessible depuis https://www.squashtest.com/community-download.

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

Note : Afin de piloter l'exécution de tests automatisés depuis Squash TM, vous avez besoin des composants suivants :
— Squash TM
— Squash Orchestrator
— Plugin Result Publisher pour Squash TM
— Plugin Squash AUTOM pour Squash TM
— Si exécution de cas de test Agilitest : l'agent OpenTestFactory pour l'environnement d'exécution des tests Agilitest

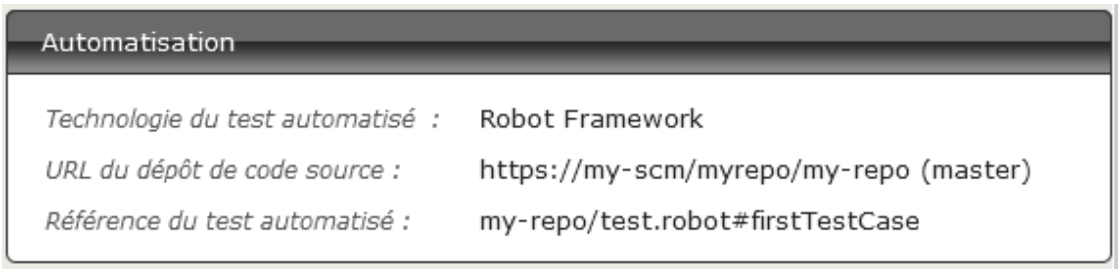
1.3.1 Automatisation d'un cas de test Squash TM

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 :

- *Agilitest*
- *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 :



- `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.` : 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 Agilitest

1. Référence du test

Pour lier un cas de test **Squash TM** à un test automatisé *Agilitest*, 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 script *ActionTestScript* à partir de la racine du projet (avec son extension `.ats`).

Avertissement : Les scripts ATS doivent impérativement se trouver dans une arborescence de fichiers de la forme `src/main/ats/*` , conformément à l'architecture classique d'un projet ATS.

2. Nature des paramètres Squash TM exploitables

Les paramètres **Squash TM** exploitables dans un script *ActionTestScript* 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

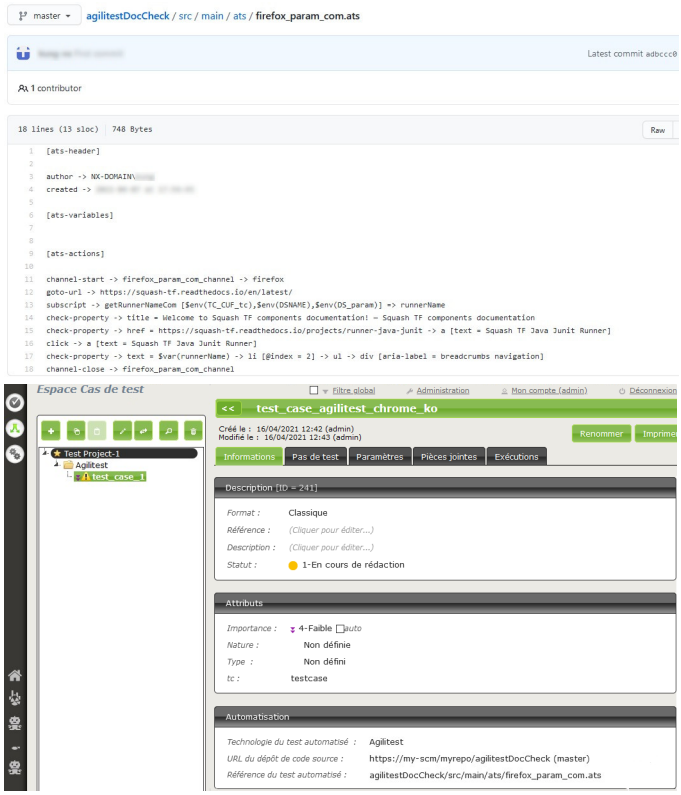
Il est possible lors de l'exécution d'un cas de test **Squash TM** automatisé avec *Agilitest* d'exploiter des paramètres **Squash TM** au sein de celui-ci.

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 existants dans le script *ActionTestScript*.

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 *Agilitest* et l'automatisation du cas de test **Squash TM** associé :



Automatisation avec Cucumber

1. Référence du test

Note : Dans le cas où un nom de scénario n'est pas spécifié, le résultat de chaque exécution du cas de test **Squash TM** est calculé en prenant en compte les résultats individuels de chaque scénario 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] # [3] # [4]
```

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`).

— `[3]` : Nom de la feature tel que renseigné dans le fichier de test *Cucumber*.
— `[4]` : Nom du scénario tel que renseigné dans le fichier de test *Cucumber*.

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*.

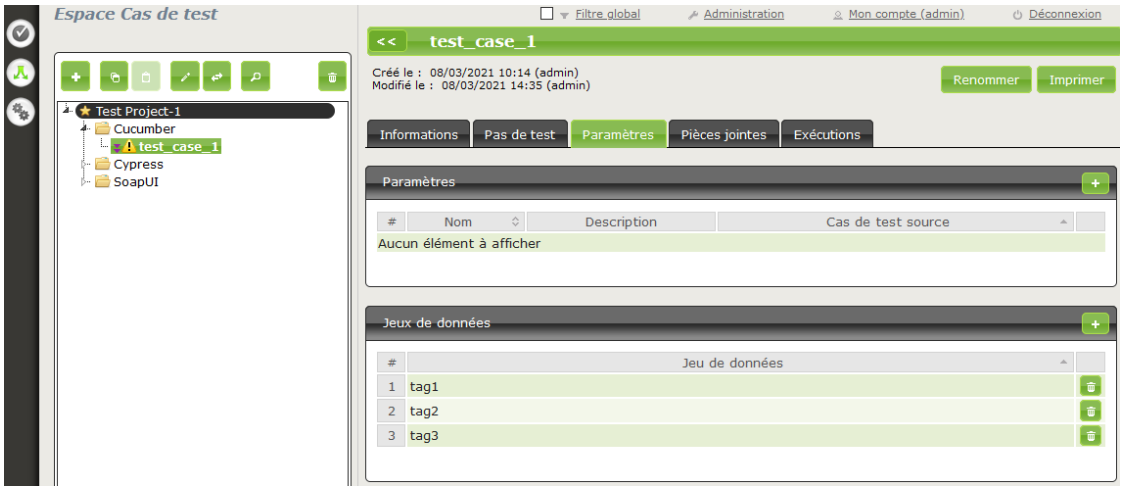
Pour cela, il faut suivre les étapes suivantes :
— Renseigner des jeux de données dans l'onglet *Paramètres* du cas de test dans **Squash TM**.
— Créer au sein de son scénario *Cucumber* autant de tableaux exemple que de jeux de données et annoter ces tableaux d'un tag correspondant au nom d'un jeu de 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 fonctionnelement 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

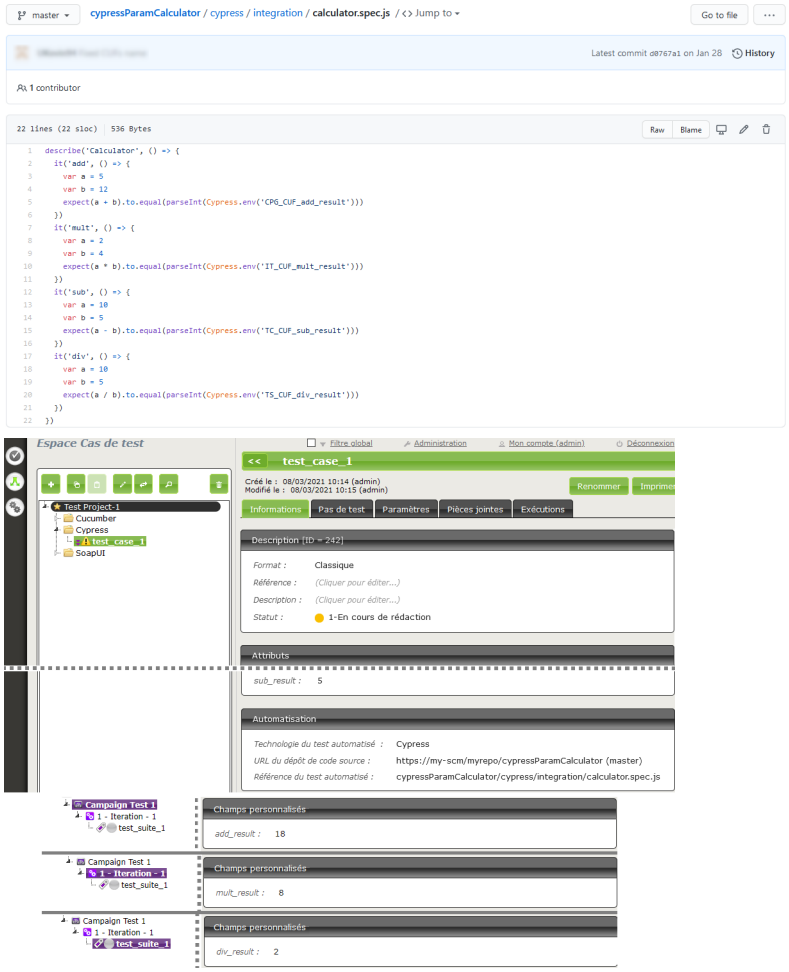
Il est possible lors de l'exécution d'un cas de test **Squash TM** automatisé avec *Cypress* d'exploiter des paramètres **Squash TM** au sein de celui-ci.

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 existants 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 :



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 :

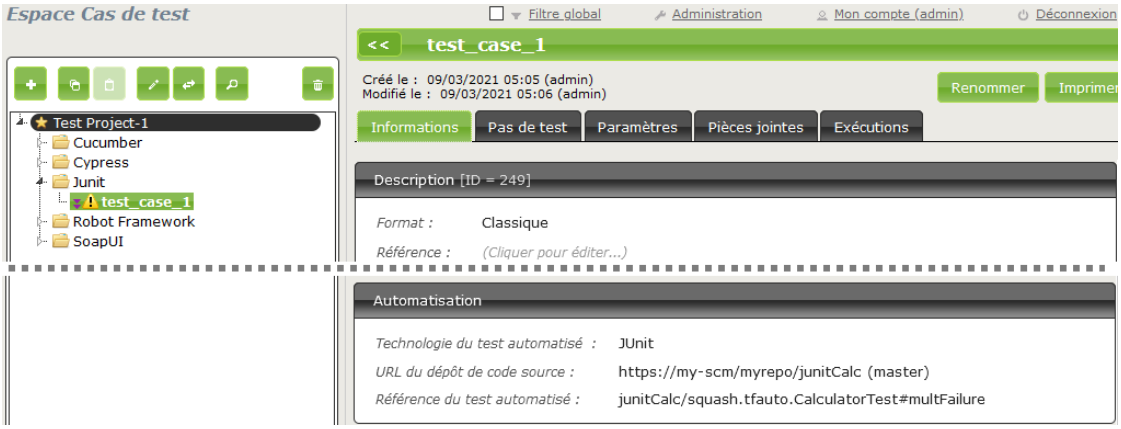
```
[1] / [2] # [3]
```

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
24
25      @Test
26      public void multFailure(){
27          int first = 2;
28          int second = 4;
29          Assertions.assertTrue((first*second)==6, "Le résultat du calcul est incorrect. " + first + " *
30      }
```



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
```

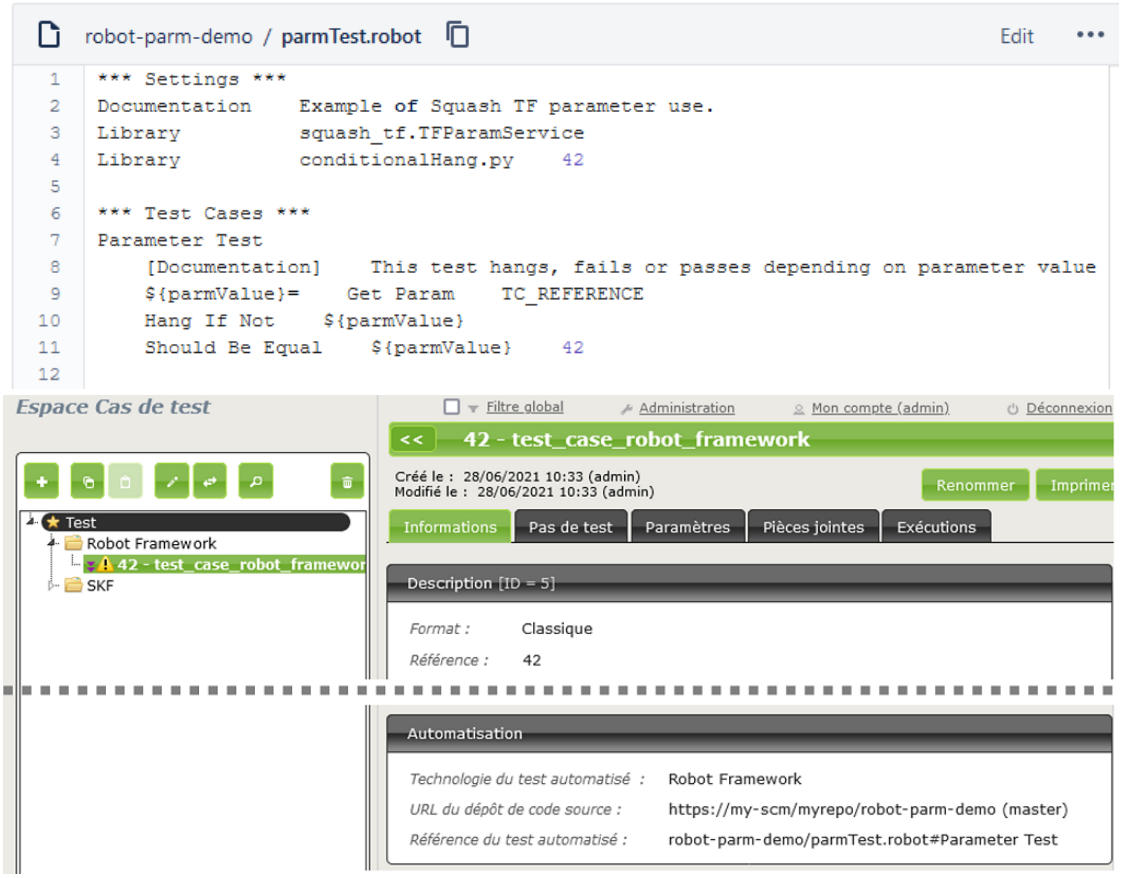
- Importer la librairie au sein du fichier `.robot` dans la section *Settings* :

```
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 :



Automatisation avec SoapUI

Référence du test

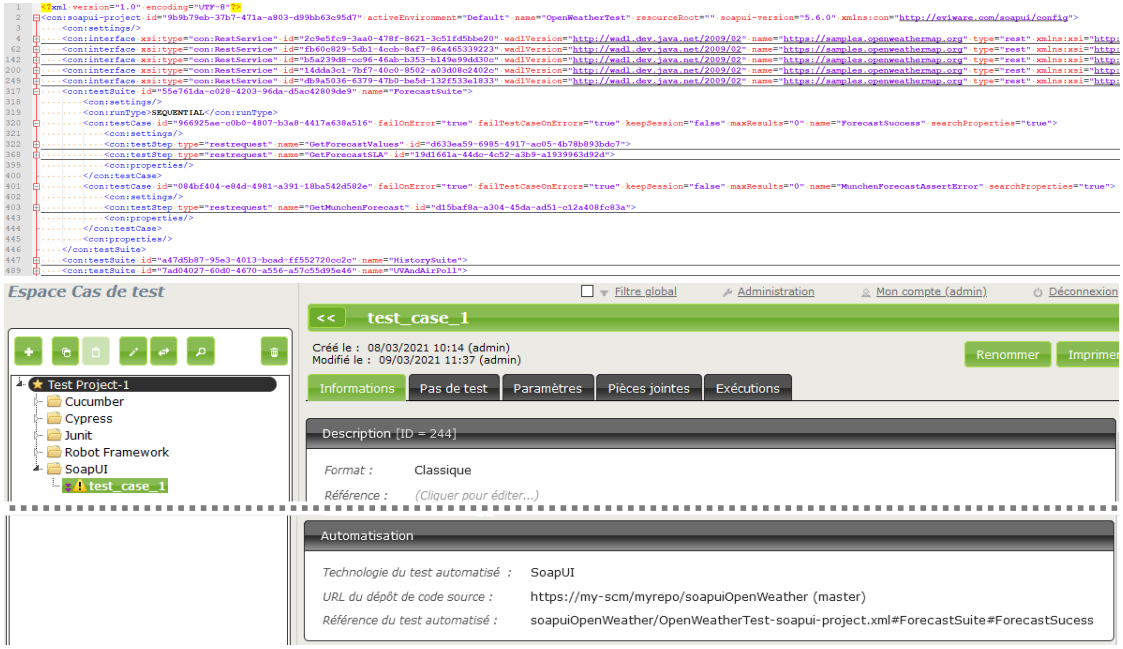
Pour lier un cas de test **Squash TM** à un test automatisé *SoapUI*, le champ *Référence* du test automatisé du bloc *Automatisation* du cas de test doit avoir la forme suivante :

```
[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.

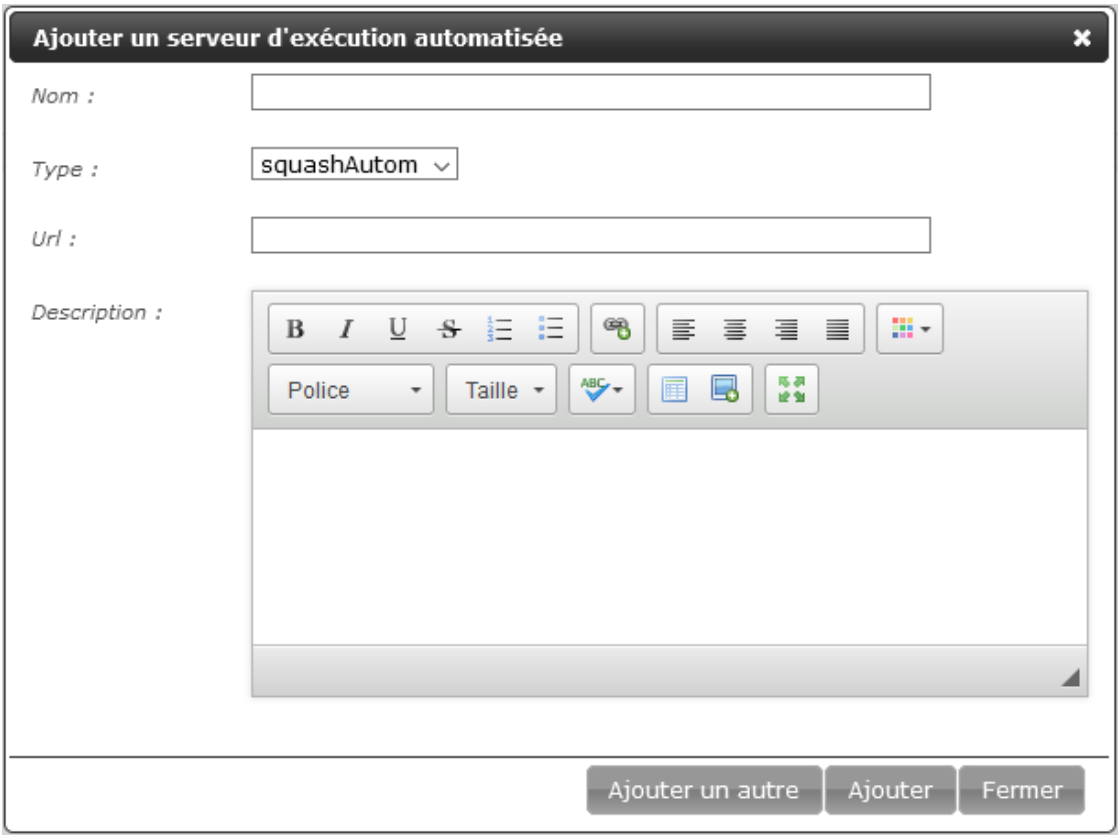
Ci-dessous un exemple de fichier de test *SoapUI* et l'automatisation du cas de test **Squash TM** associé :



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

Déclaration du serveur Squash Orchestrator

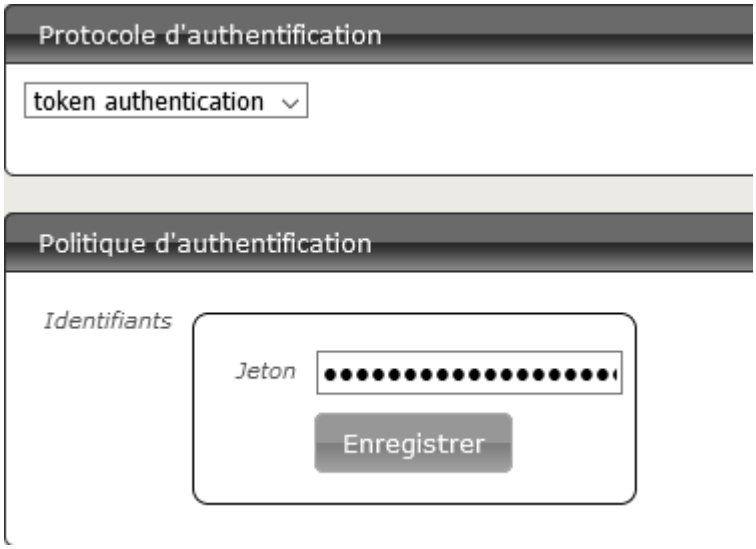
Afin de lancer un plan d'exécution manuellement depuis **Squash TM**, il est nécessaire de déclarer le serveur **Squash Orchestrator** qui exécutera les tests automatisés dans les environnements adaptés. Cette déclaration se fait dans la section *Serveurs d'exécution automatisée* de l'espace *Administration*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



— `Nom` : Le nom du serveur tel qu'il apparaitra 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.

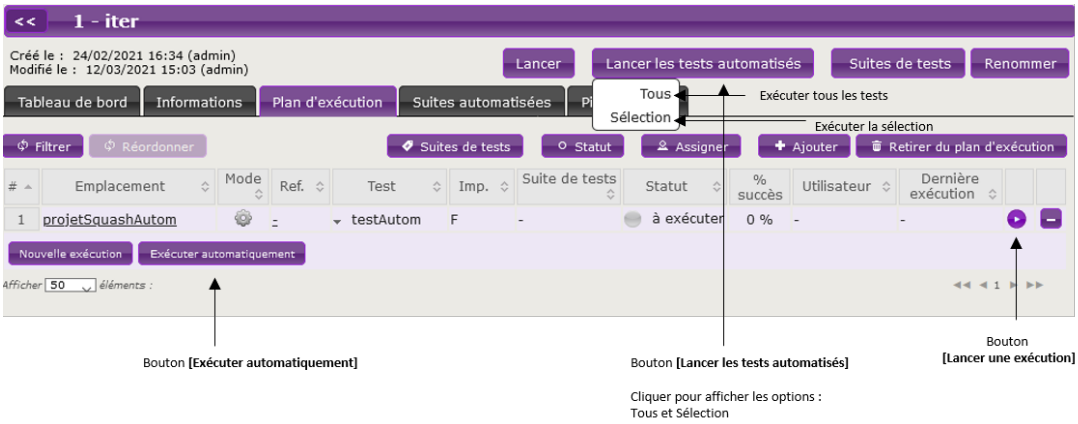


Note : La présence d'un token est obligatoire pour la bonne exécution de tests automatisés depuis **Squash TM**. Dans le cas où l'authentification au serveur d'automatisation n'est pas requise par ce dernier, il faut quand même renseigner une valeur quelconque de token dans **Squash TM**

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 :



— 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.

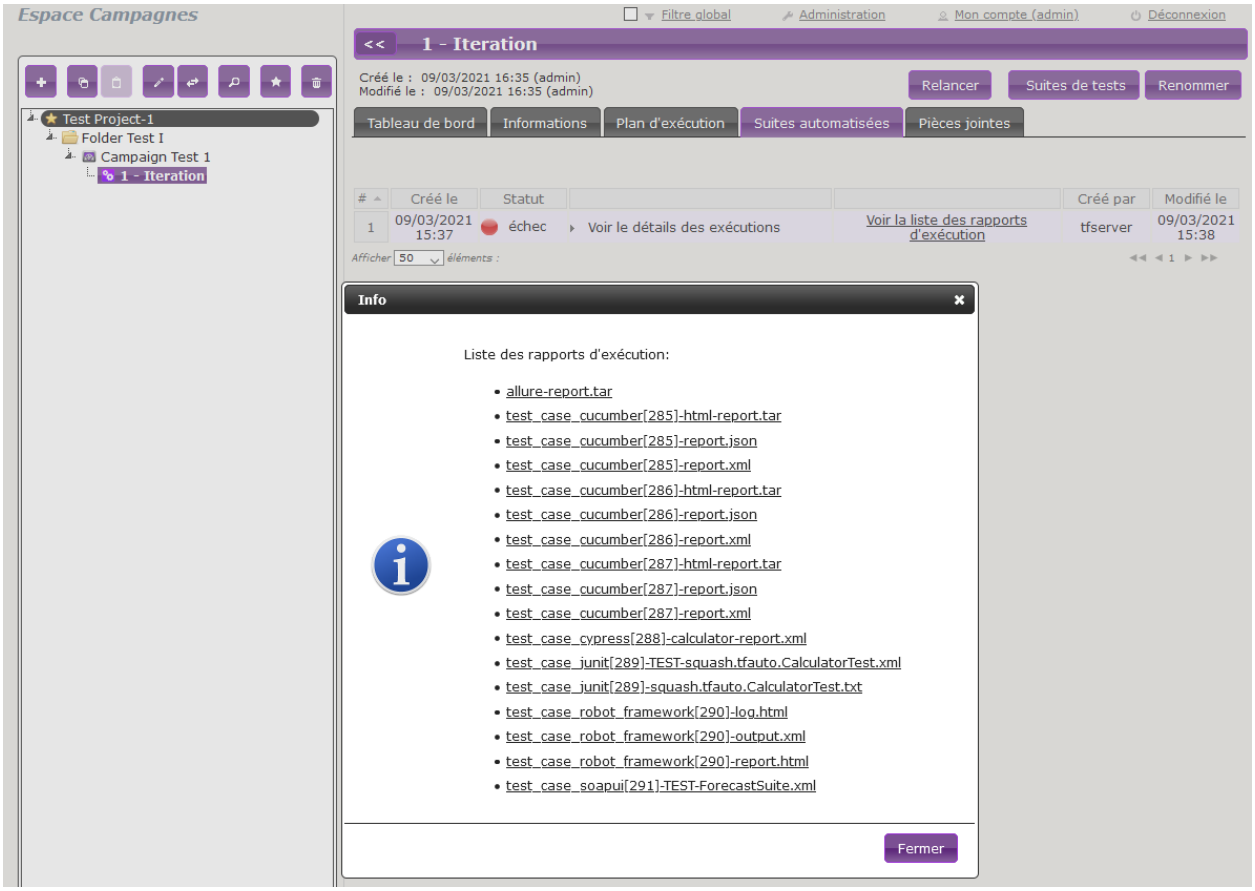
1.3.3 Remontées de résultats après exécution d'un plan d'exécution Squash TM

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 :



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*.

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 :

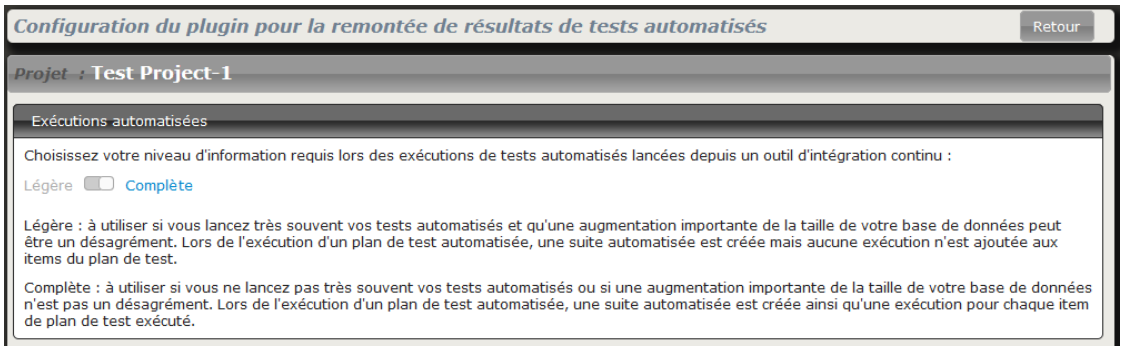
— Création d'une nouvelle exécution pour chaque ITPI qui a été exécuté.

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.

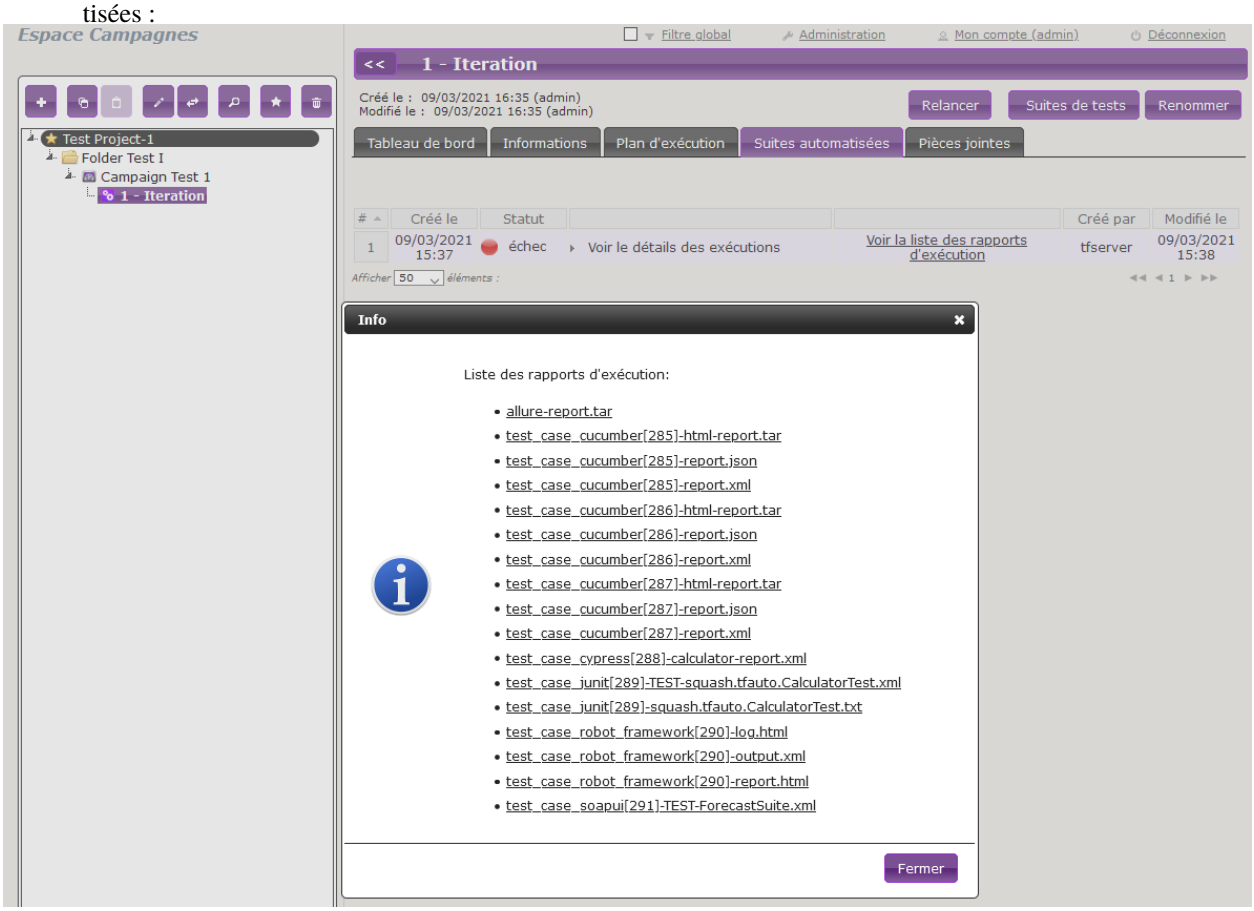
Le choix du type de remontée se fait par projet en accédant à la configuration du plugin **Squash TM Result Publisher** depuis l'onglet *Plugins* de la page de consultation d'un projet :



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 :



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*.

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

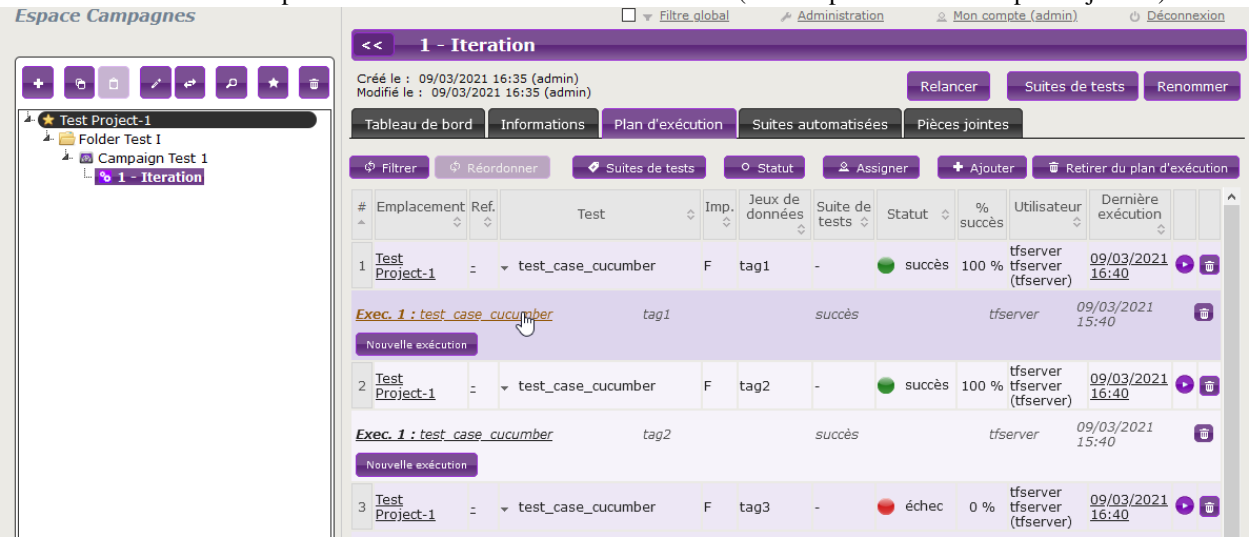
Cependant, voici ce qu'il ne se passe pas :

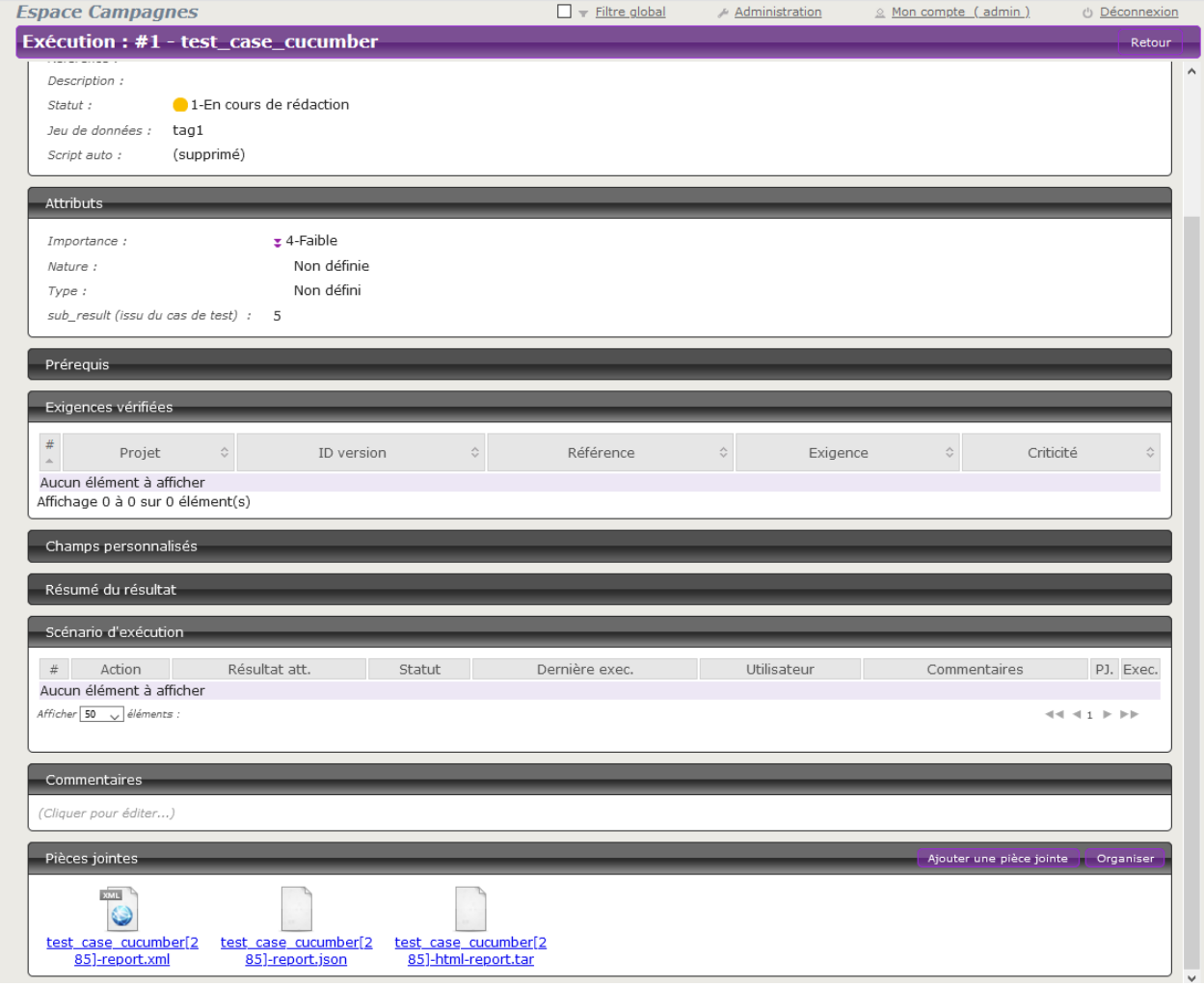
— Création d'une nouvelle exécution pour chaque ITPI qui a été exécuté.

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).





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*.

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 offertes par la version *1.0.0.RELEASE* de **Squash AUTOM**.

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

— **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.

— **Agent OpenTestFactory** : cet agent permet des communications via le protocole HTTP entre un **Squash Orchestrator** et un environnement d'exécution de tests. Il est actuellement nécessaire pour l'exécution de tests Agilitest.

— **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**.

Avertissement : Ce site est obsolète, la documentation de Squash AUTOM et Squash DEVOPS est désormais https://autom-devops-fr.doc.squashtest.com/.

CHAPITRE 2

Squash DEVOPS

2.1 Guide d'Installation

2.1.1 Squash Generator Service

Présentation

Ce micro-service permet la récupération de plans d'exécution de tests automatisés **Squash TM** pour exécution au sein d'un PEaC.

Installation

Le micro-service est inclus dans l'image Docker de **Squash Orchestrator**. Pour plus de détails sur l'installation de **Squash Orchestrator**, consultez la *section dédiée* de la documentation.

Ce micro-service existe en version **Squash DEVOPS Community** et **Squash DEVOPS Premium** et marche conjointement avec le plugin Test Plan Retriever dans la même version (Community ou Premium).

Pour exécuter l'image Docker de **Squash Orchestrator** avec la version **Squash DEVOPS Premium** du micro-service **Squash Generator**, le paramètre `-e SQUASH_LICENCE_TYPE=premium` est à préciser dans la commande `docker run` de déclenchement de **Squash Orchestrator**.

2.1.2 Plugin Test Plan Retriever pour Squash TM

Présentation

Ce plugin permet la récupération de plans d'exécution de tests automatisées **Squash TM** par une instance de **Squash Orchestrator**.

Installation

Le plugin existe en version **Community** (*squash.tm.rest.test.plan.retriever.community-1.0.0.RELEASE.jar*) librement téléchargeable ou **Premium** (*squash.tm.rest.test.plan.retriever.premium-1.0.0.RELEASE.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* ou supérieure de **Squash TM**.

2.1.3 Plugin Squash DEVOPS pour Jenkins

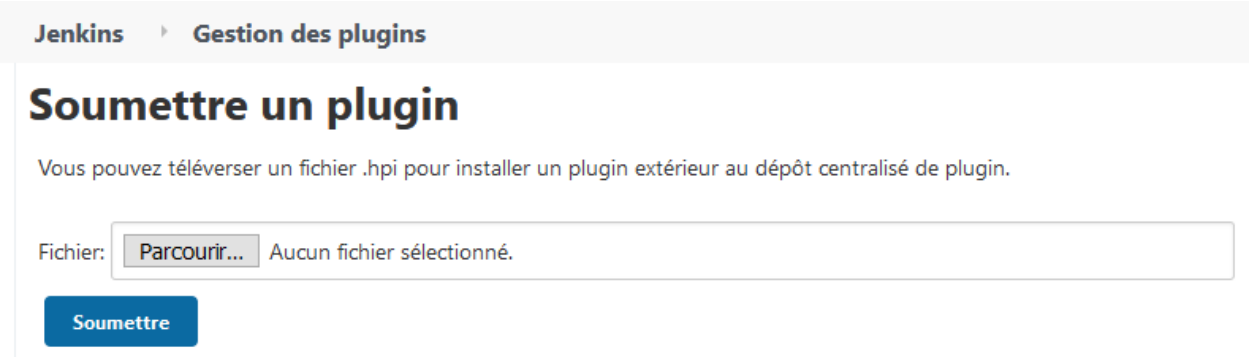
Présentation

Ce plugin Jenkins permet d'intégrer facilement des appels à **Squash Orchestrator** au sein d'un pipeline.

Installation

Le plugin est sous la forme d'un fichier `.hpi` (*squash-devops-1.0.0.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* :



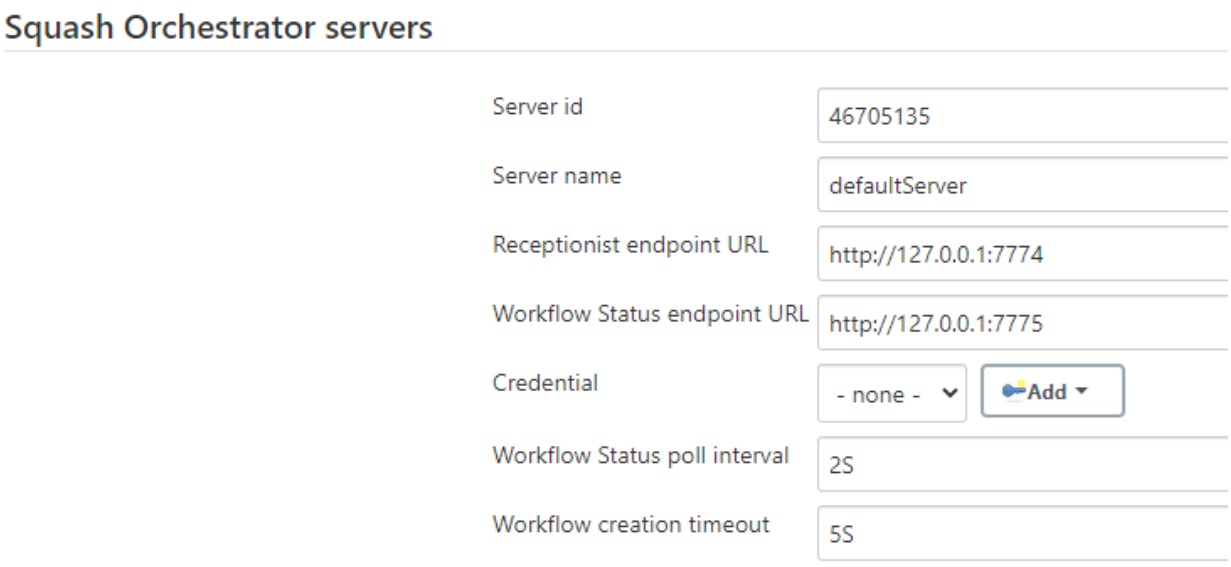
Avertissement : Ce plugin est compatible avec une version *2.164.1* ou supérieure de *Jenkins*.

2.2 Appel au Squash Orchestrator depuis un pipeline Jenkins

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* :

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



— `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(
```

(suite sur la page suivante)

(suite de la page précédente)

```
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

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).

Note : Pour le bon fonctionnement de cette fonctionnalité, les plugins Test Plan Retriever et Result Publisher doivent être installés sur l'instance Squash TM ciblé.

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'informations 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.RELEASE* de **Squash DEVOPS**.

Cette version *1.0.0.RELEASE* 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*.

Avertissement : Ce site est obsolète, la documentation de Squash AUTOM et Squash DEVOPS est désormais https://autom-devops-fr.doc.squashtest.com/.

CHAPITRE 3

FAQ

3.1 FAQ : Généralités autour de Squash AUTOM et Squash DEVOPS

Cette FAQ cherche à répondre à des questions sur le pourquoi de **Squash AUTOM** et **Squash DEVOPS**, ce que cela implique pour **Squash TF** et comment va s'opérer la transition de **Squash TF** à **Squash AUTOM** et **Squash DEVOPS**.

3.1.1 Pourquoi deux nouveaux produits Squash AUTOM et Squash DEVOPS ?

La création de Squash AUTOM et Squash DEVOPS est le résultat d'une réflexion sur l'évolution des pratiques d'automatisation (essor des pratiques du CI/CD et du DevOps, utilisation de plus en plus démocratisée de la conteneurisation, multiplication des outils d'intégration) et sur comment la suite logicielle Squash pouvait être en phase avec celles-ci.

Il en est ressorti que Squash TF présentait des limites, notamment architecturales, pour son adoption au sein des principes DevOps.

C'est pourquoi nous avons décidé de développer un nouvel outil destiné à la gestion de l'exécution des tests automatisés respectant les principes suivants :

— Architecture micro-service, notamment pour des raisons de déploiement et d'exploitabilité en environnement DevOps.
— Séparation entre les fonctionnalités permettant d'automatiser (à destination des testeurs et automaticiens) et celles permettant d'intégrer les tests automatisés (pour le gestionnaire de pipeline) au sein de l'usine DevOps. Cela a donc donné naissance à 2 produits nommés Squash AUTOM et Squash DEVOPS.
— Suppression de l'adhérence avec Squash TM de manière à rendre ces deux produits indépendants de celui-ci.

3.1.2 Quel est le modèle de Squash AUTOM et Squash DEVOPS ?

Le modèle retenu est un modèle « open core ».

Ce modèle, qui est le même que Squash TM, met à disposition deux versions :

— Une version Community gratuite composée d'un cœur open source et de modules freemium. Cette version est pleinement fonctionnelle (non bridée).
— Une version commerciale, avec souscription annuelle, composée de la version Community et de plugins commerciaux. Elle apporte des fonctionnalités supplémentaires à valeur ajoutée, mais non indispensables, ainsi que le support.

3.1.3 Squash AUTOM et Squash DEVOPS peuvent-ils s'utiliser sans Squash TM ?

Oui.

Notre but est que les deux produits apportent également de la valeur aux sociétés ou projets n'utilisant pas Squash TM :

— L'utilisation de Squash AUTOM « seul » permet ainsi d'unifier/d'homogénéiser l'usage des différents automates (Selenium, Cypress, SoapUI, Appium…) et des différents studios (Robot Framework, Cucumber, UFT, Agilitest…) tout en générant un format de reporting commun (type Allure).
— L'utilisation de Squash DEVOPS « seul » permet d'orchestrer l'ensemble des tests automatisés, de les intégrer au pipeline DevOps (CI/CD) puis de poster les résultats vers les destinataires (le pipeline lui-même, l'outil de patrimoine de test ou le framework de reporting et d'agrégation des résultats de test).

3.1.4 Est-ce que de nouvelles fonctionnalités pour Squash TF arriveront dans le futur ?

Non, il n'y aura plus de nouvelles fonctionnalités développées pour Squash TF.

Nous vous encourageons à faire la transition de Squash TF à Squash AUTOM pour l'exécution de votre patrimoine de tests automatisés afin de profiter de l'ensemble des nouvelles fonctionnalités proposées par Squash.

Néanmoins, les éléments de Squash TF resteront accessibles en téléchargement. De même, les répertoires open source resteront accessibles.

3.1.5 Le support pour Squash TF s'arrête-t-il avec la sortie de Squash AUTOM et Squash DEVOPS ?

Non.

Nous continuerons à assurer du support sur Squash TF via le forum Squashtest et, pour les clients de l'offre commerciale Squash AUTOM, *via* notre service support.

3.1.6 Mon patrimoine de tests automatisés, exécutés jusque-là avec Squash TF, doit-il être modifié pour être utilisé avec Squash AUTOM ?

Non.

Les scripts/tests automatisés que vous exécutiez *via* Squash TF sont exploitables par Squash AUTOM sans modification de ceux-ci.

3.1.7 Puis-je exécuter des tests SKF avec Squash AUTOM et Squash DEVOPS ?

Pas dans la version 1.0.0.RELEASE.

Le support des tests SKF sera disponible dans une version postérieure avant la fin du deuxième trimestre 2021.

3.1.8 Que dois-je faire dans Squash TM pour lancer mes plans d'exécutions automatisées avec Squash AUTOM au lieu de Squash TF ?

Il est nécessaire de créer un lien entre un cas de test Squash TM et votre test automatisé conformément à la documentation de Squash AUTOM.

Cette action est quasi instantanée et peut se faire en masse pour vos cas de test Squash TM Gherkin ou BDD exploitant le plugin Git. Pour les autres cas de test, une action sur chaque cas de test sera nécessaire conformément à la documentation de Squash AUTOM.

L'action de lien entre un cas de test Squash TM et un test automatisé pour une exécution avec Squash AUTOM est différente de celle pour une exécution avec Squash TF.

3.1.9 Puis-je mélanger dans un même plan d'exécution des cas de tests automatisés exécutés par Squash TF et des cas de tests exécutés par Squash AUTOM ?

Oui.

Afin de faciliter la transition, il est parfaitement possible d'avoir, au sein d'un même plan d'exécution Squash TM, des cas de tests issus d'un projet exploitant Squash TF et des cas de tests issus d'un projet exploitant Squash AUTOM.

3.1.10 Dois-je forcément avoir un serveur Jenkins pour pouvoir exécuter mes tests automatisés depuis Squash TM via Squash AUTOM ?

Non.

Les jobs Jenkins spécifiques nécessaires pour l'exécution de tests automatisés depuis Squash TM via Squash TF ne sont plus un prérequis pour une exécution depuis Squash TM via Squash AUTOM.

Avec Squash AUTOM, l'exécution est assurée par le Squash Orchestrator, un composant spécifique de Squash AUTOM.

3.1.11 Puis-je lancer mes plans d'exécutions automatisées Squash TM depuis un pipeline Jenkins avec Squash AUTOM et SQUASH DEVOPS ?

Oui.

L'exécution d'un plan d'exécution Squash TM depuis un pipeline Jenkins est une nouveauté de Squash DEVOPS par rapport à Squash TF et nécessite la mise en place de jobs suivant les indications de la documentation Squash DEVOPS.

Cette section présente un ensemble de FAQ autour de **Squash AUTOM** et **Squash DEVOPS**.

Les FAQ disponibles sont les suivantes :

— *Généralités* : FAQ sur des généralités autour de **Squash AUTOM** et **Squash DEVOPS** et les changements par rapport à **Squash TF**