Squash Autom & Squash Devops

Version 1.0.0-alpha1

squashtest

mars 12, 2021

Table des matières

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.

CHAPITRE 1

Squash AUTOM

1.1 Guide d'Installation

1.1.1 Squash Orchestrator

Squash Orchestrator est disponible sous forme d'une image Docker sur DockerHub (squashtest/squash-orchestrator :1.0.0.alpha1).

Pour la méthode de déploiement, consulter la documentation de **Squash Orchestrator** (Squash Orchestrator Documentation – 1.0.0.alpha1 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.alpha1.jar) librement téléchargeable ou Premium (squash.tm.rest.result.publisher.premium-1.0.0.alpha1.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 1.22.1.RELEASE 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.

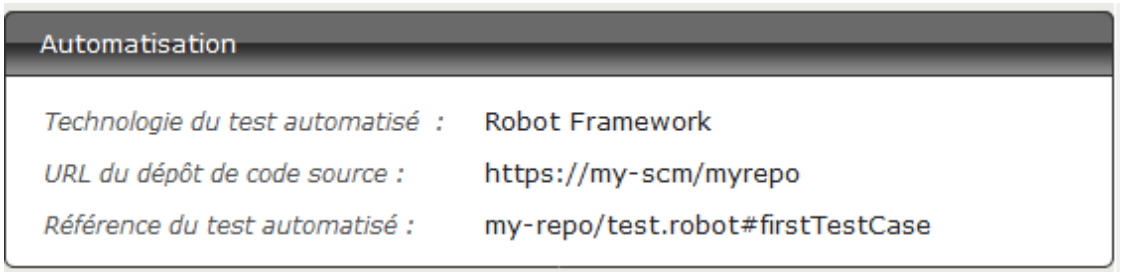
Retrouvez plus d'informations sur la rédaction d'un PEAC au sein de la documentation de **Squash Orchestrator** (Squash Orchestrator Documentation – 1.0.0.alpha1 version.pdf) accessible depuis https://www.squashtest.com/community-download.

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

1.3.1 Automatisation d'un cas de test Squash TM

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* et *Junit* sont fonctionnels.
— `URL du dépôt de code source` : L'adresse du dépôt de source où se trouve le projet.
— `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* et *Junit* 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*).

Avertissement : Un problème connu de la version est que le remplissage automatique par Squash TM ne met pas le nom du repo en tête de la référence du test automatisé. Il est nécessaire et doit donc être ajouté manuellement par l'utilisateur. Le nom correspond à la dernière partie de l'URL du dépôt de code source.

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.

Spécificités pour l'automatisation suivant le framework d'automatisation

Automatisation avec Robot Framework

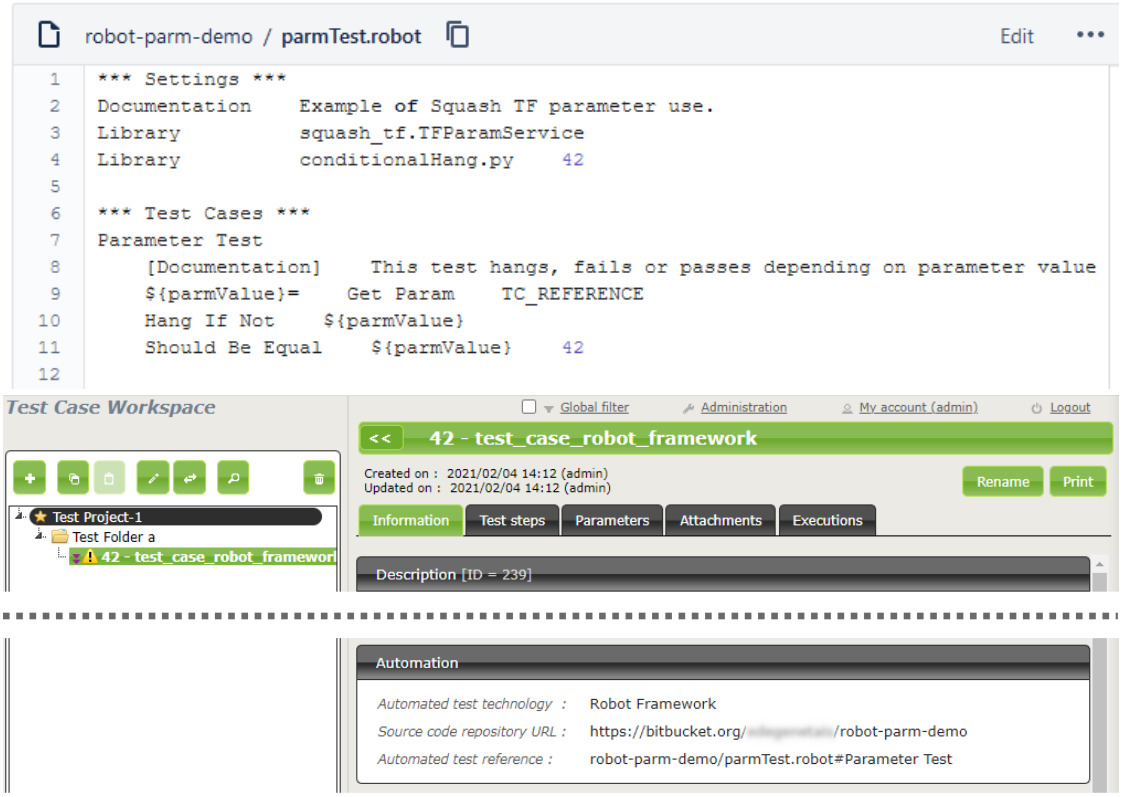
Pour lier un cas de test **Squash TM** à un test automatisé *Robot Framework*, le champ *Référence* du test automatisé du bloc *Automatisation* du cas de test doit avoir la forme suivante :

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

Avec :

- `[1]` : Nom du projet sur le dépôt de source.
- `[2]` : 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`.

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



Automatisation avec JUnit

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.

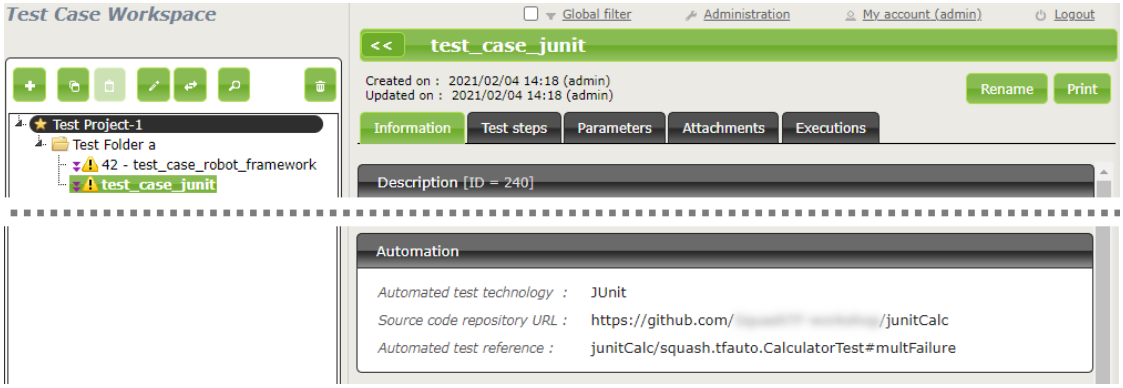
Ci-dessous un exemple de classe de test et l'automatisation du cas de test **Squash TM** associé :

```java
package squash.tfauto;

import org.junit.jupiter.api.Assertions;
import org.junit.jupiter.api.DisplayName;
import org.junit.jupiter.api.Test;

@DisplayName("Calculator")
public class CalculatorTest {

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



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

Cette fonctionnalité n'est pas disponible dans la version 1.0.0.alpha1.

Il est néanmoins possible de déclencher un plan d'exécution **Squash TM** depuis un pipeline *Jenkins*. Référez-vous au guide utilisateur de *Squash DEVOPS*.

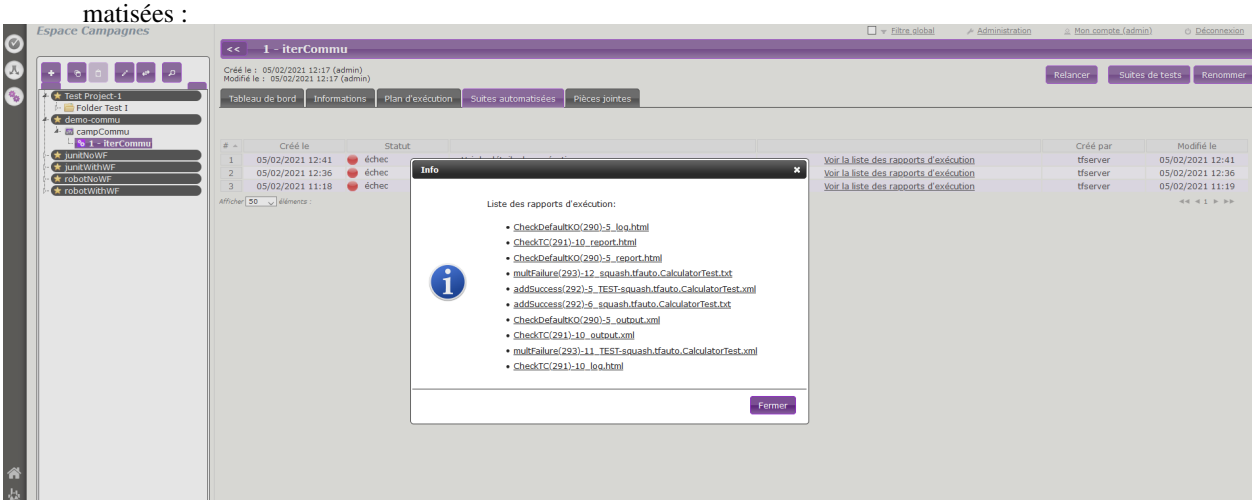
1.3.3 Remontées de résultats après exécution d'un plan d'exécution Squash TM

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 Test), les informations suivantes sont mises à jour :

- Mise à jour du statut des ITPI.
- Mise à jour du statut de la suite automatisée.
- Les rapports d'exécution des différents ITPI sont accessibles depuis l'onglet de consultation des suites automatisées :



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

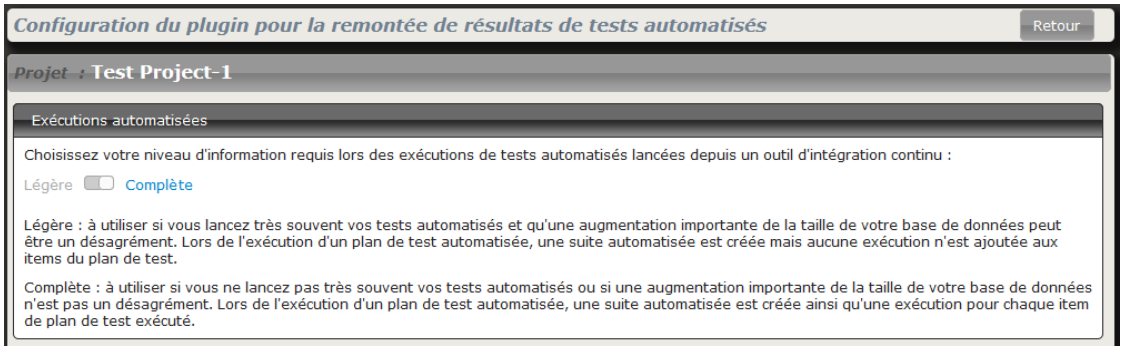
- Création d'une nouvelle exécution pour chaque ITPI qui a été exécuté.

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.

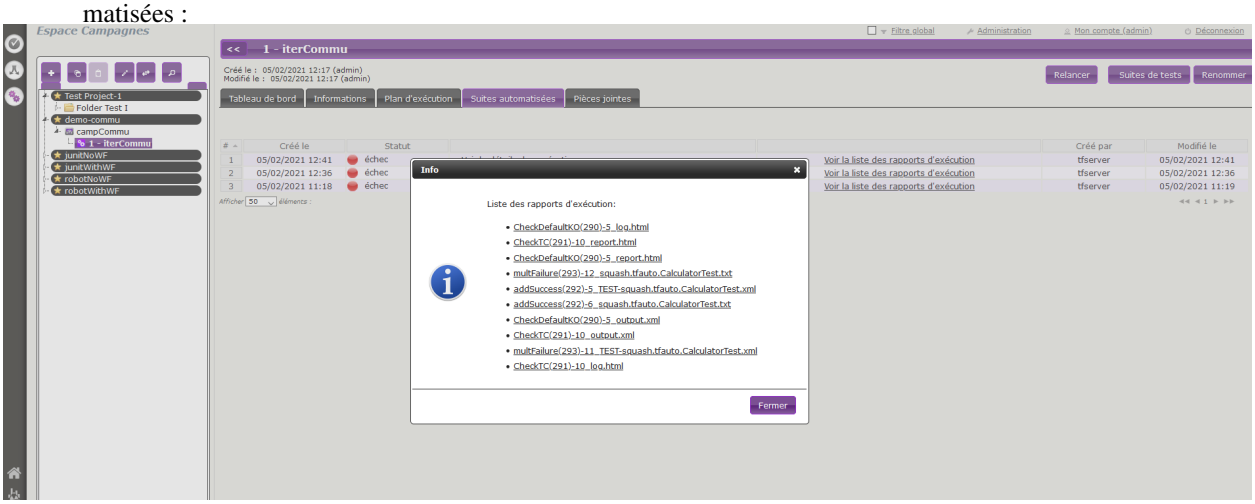
Le choix du type de remontée se fait par projet en accédant à la configuration du plugin **Squash TM Result Publisher** depuis l'onglet Plugins de la page de consultation d'un projet :



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 Test) :

— Mise à jour du statut des ITPI.
— Mise à jour du statut de la suite automatisée.
— Les rapports d'exécution des différents ITPI sont accessibles depuis l'onglet de consultation des suites automatisées :



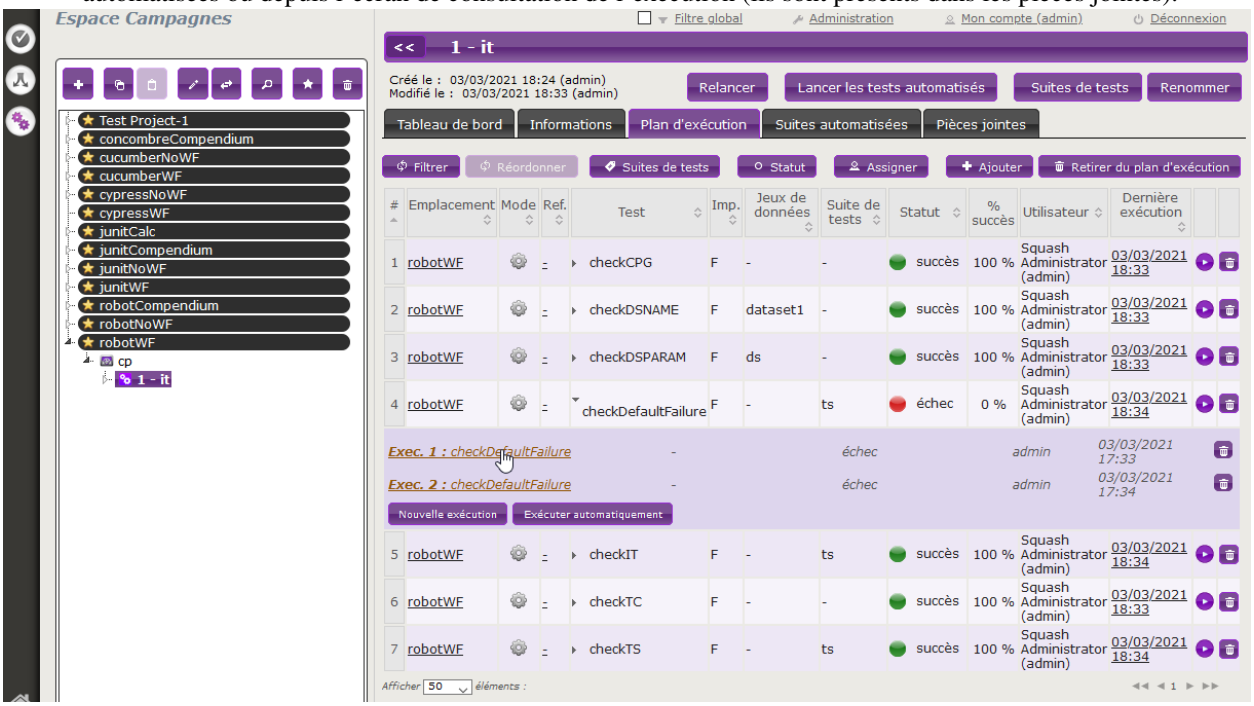
Cependant, voici ce qu'il ne se passe pas :

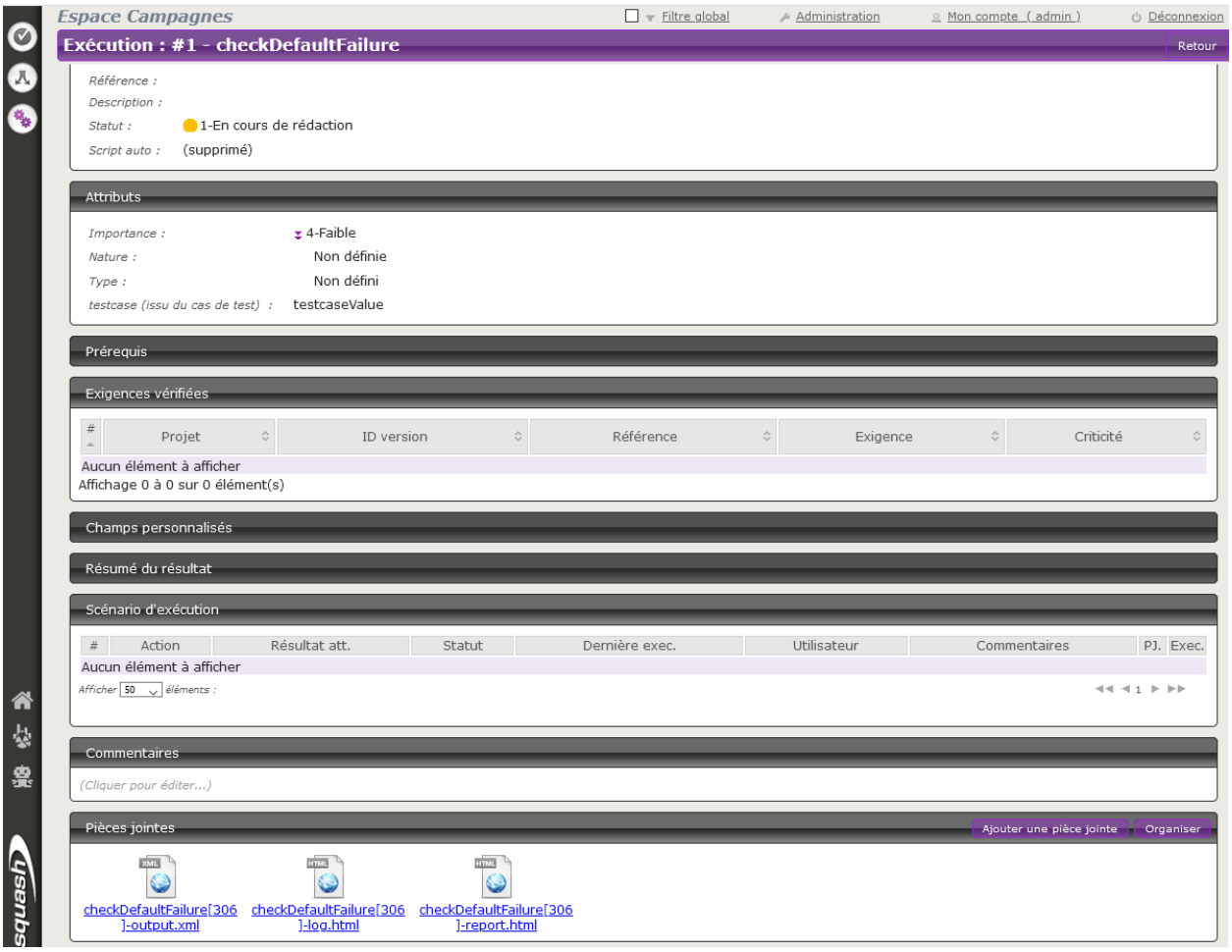
— Création d'une nouvelle exécution pour chaque ITPI qui a été exécuté.

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 Test) :

- Mise à jour du statut des ITPI.
- Création d'une nouvelle exécution pour chaque ITPI.
- Mise à jour du statut de la suite automatisée.
- 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).





Ce guide vous présente les possibilités offerte par la version 1.0.0.alpha1 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.alpha1 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.

CHAPITRE 2

Squash DEVOPS

2.1 Guide d'Installation

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.alpha1* 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.alpha1.jar*) librement téléchargeable ou **Premium** (*squash.tm.rest.test.plan.retriever.premium-1.0.0.alpha1.jar*) accessible sur demande.

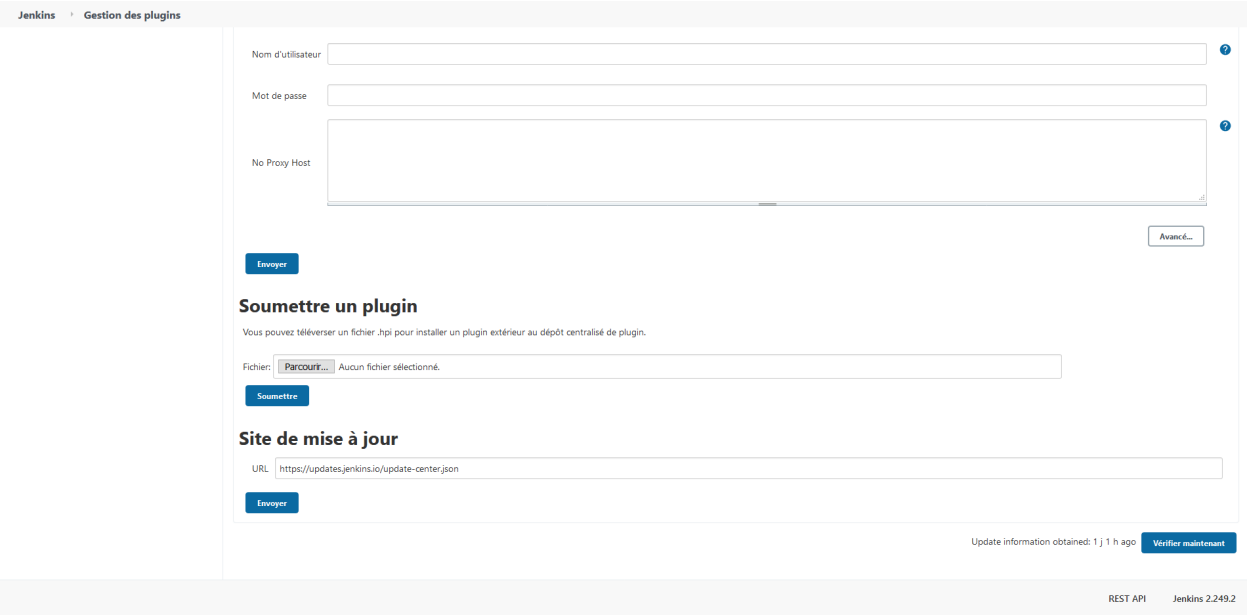
Pour l'installation, merci de vous reporter au protocole d'installation d'un plugin **Squash TM** (https://sites.google.com/a/henix.fr/wiki-squash-tm/installation-and-exploitation-guide/2—installation-of-squash-tm/7—jira-plug-in).

Avertissement : Ce plugin est compatible avec une version 1.22.1.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.alpha1.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 :



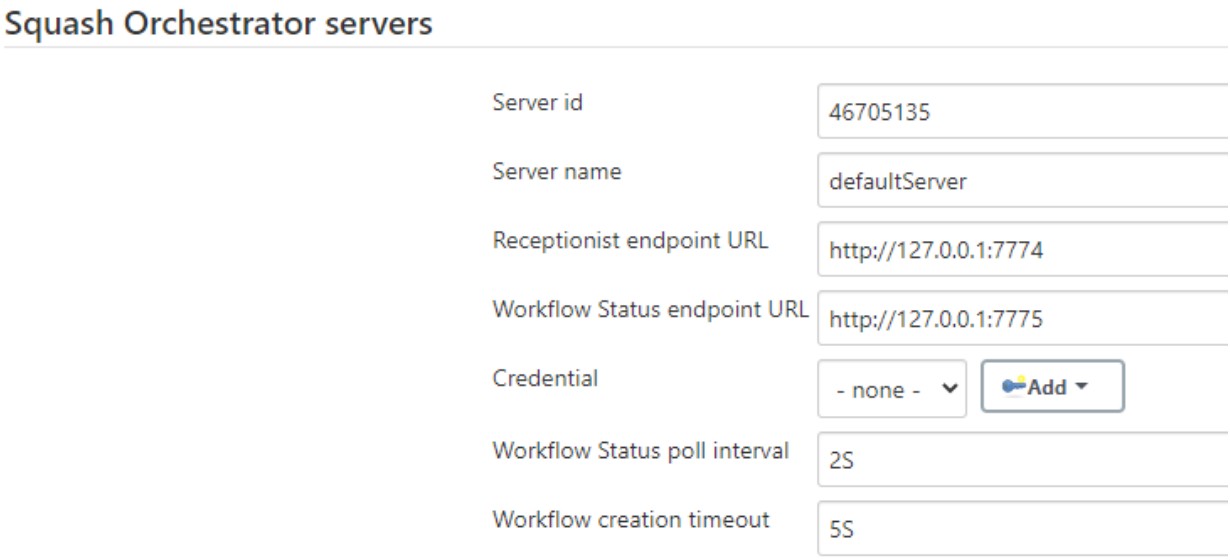
Avertissement : Ce plugin est compatible avec une version 2.164.1 ou supérieure de Jenkins.

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 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 = runSquashTFWorkflow(
        workflowPathName:'testWorkflow.json',
        workflowTimeout: '20S',
        serverName:'defaultServer'
        )
        echo "We just ran The Squash Orchestrator workflow $workflow_id"
   }
}
```

La méthode *runSquashTFWorkflow* 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).

2.3.1 Prérequis

Afin de pouvoir récupérer un plan d'exécution **Squash TM** depuis un PEAC, vous avez besoin d'avoir effectuer 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",
```

(suite sur la page suivante)

(suite de la page précédente)

```
            "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 test 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 automatisée Robot Framework

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 *Robot Framework*.

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*

Utilisation de paramètres Squash TM dans un cas de test Robot Framework

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 :

— 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>
```

Vous verrez ci-dessous un exemple de cas de test *Robot Framework* exploitant les paramètres **Squash TM** :

robot-parm-demo / parmTest.robot Edit

```
*** Settings ***
Documentation    Example of Squash TF parameter use.
Library          squash_tf.TFParamService
Library          conditionalHang.py    42

*** Test Cases ***
Parameter Test
    [Documentation]    This test hangs, fails or passes depending on parameter value
    ${parmValue}=    Get Param    TC_REFERENCE
    Hang If Not    ${parmValue}
    Should Be Equal    ${parmValue}    42

```

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 DEVOPS 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.alpha1 de **Squash DEVOPS**.

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.alpha1 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 **Squash AUTOM** 1.0.0.alpha1 (téléchargeable depuis https://www.squashtest.com/community-download) 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*.