VITAM - Scénarios de test

Version 0.26.1

VITAM

nov. 22, 2017

Table des matières

CHAPITRE 1

Objectif du document

Ce document a pour objectif de présenter les différentes méthodes et outils de test pour pouvoir tester au maximum les fonctionnalités offertes par la solution logicielle Vitam, que ce soit via ses API ou en passant par un outillage de tests automatisés.

CHAPITRE 2

Outils de tests

Divers outils ont été mis en place afin de vérifier chaque aspect de la solution logicielle VITAM :

- Les tests manuels permettent de tester un large spectre de fonctionnalités de la solution logicielle Vitam lors des développements.
- Les tests automatiques permettent de vérifier de manière régulière qu'une régression n'est pas survenue et que tout fonctionne correctement (chapitre 4).

Plusieurs documents complémentaires sont à disposition :

- La documentation des Tests de Non Régression (TNR) se trouve dans : doc/fr/configuration-tnr/configuration.rst
- Le manuel d'intégration applicative qui présente la manière d'interroger le DSL est présent dans ce même document doc/fr/configuration-tnr/configuration.rst
- Le tableau du cahier de tests manuels se trouve dans l'outil Jalios (espace livraison)

Administration des collections

L'administration des collections est accessible dans l'IHM recette via le menu éponyme. Cela permet de purger les référentiels, les journaux, les unités archvistiques et groupes d'objest et les contrats par collection (au sens MongoDB) ou pour la totalité des collections présentent dans cette IHM.

CHAPITRE 3

Tests Manuels

Les tests manuels peuvent être effectués :

- A l'aide du cahier de tests manuels
- Via l'IHM recette, qui permet de lancer des requêtes DSL

3.1 Cahier de tests manuels

Le cahier de test se présente sous forme de tableur. Il répertorie méticuleusement chaque cas de test possible.
Le tableau contient :

- Le titre explicite du cas de test
- L'itération à laquelle le test se raccroche
- La liste des User Stories qui traitent ce cas de test
- Le nom de l'activité, nom associé au code Story Map
- Le Code Story Map, c'est-à-dire le code attribué à ce sujet (entrée, accès, stockage, etc.)
- Le Use Case ou déroulement du test étape par étape
- IHM / API, spécifie pour quelle interface le test est dédié
- Le ou les jeux de tests associés

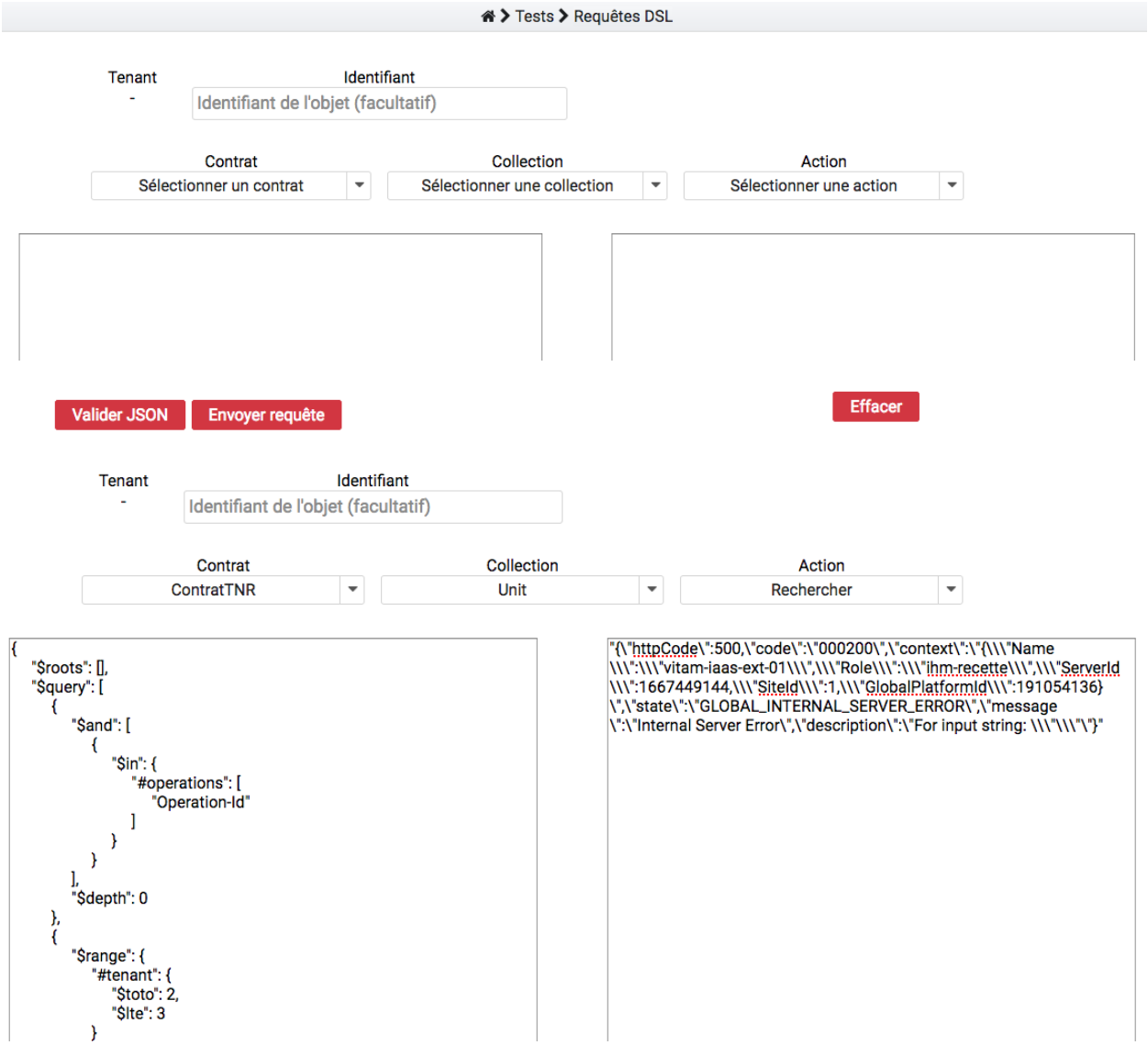
3.2 Requêtes DSL

Il est possible de lancer des requêtes DSL via l'IHM de recette depuis le menu "Tests / Tests requêtes DSL", sans besoin de certificat. Cela permet de tester de manière simple et rapide des requêtes DSL.

Un formulaire permet de gérer plusieurs variables. Un tenant doit être sélectionné aupréalable au niveau du menu. Au niveau du formulaire, il faut choisir :

- contrat d'accès sur lequel lancer le test
- la collection relative à la requête
- l'action à tester
- un identifiant (facultatif)

La requête est ensuite écrite dans le champ texte de gauche. Le bouton “Valider JSON” permet de vérifier sa validité avant de l’envoyer. Un clic sur le bouton “Envoyer requête” affiche les résultats sous format JSON dans le champ texte de droite.



CHAPITRE 4

Tests Automatisés

4.1 Tests fonctionnels

** Cucumber **

Cucumber est un outil de tests fonctionnels, il est accessible via l'IHM de recette dans le menu "Tests / Tests fonctionnels". Ces tests sont effectués via des ordres écrits avec des phrases simples, ce qui offre une grande variété de combinaisons.

Il existe une liste de contextes et de fonctions disponibles. Il s'agit ensuite de les associer et les manipuler afin de créer son propre test.

Les résultats sont retournés sous forme de tableau

** Tests de stockage **

Ces tests permettent de vérifier qu'un objet est bien stocké plusieurs fois sur la plateforme, afin d'assurer sa pérennité.

Ce test vérifie :

- Le tenant sur lequel est stocké l'objet
- Le nom de l'objet stocké
- La strategie de stockage
- La liste des stratégies où est stocké l'objet
- La présence de l'objet dans ces stratégies

4.2 Séquencement de tests

Un fichier contient une liste des TNR qui seront lancés de manière séquencée afin de réaliser et tester un scénario complet.

CHAPITRE 5

Cucumber (exemples)

Voici quelques exemples de scénarios de tests réalisables avec Cucumber.

5.1 Collection Unit

Fonctionnalité Recherche avancée

Scénario Recherche avancée d'archives – cas OK d'une recherche multicritères croisant métadonnées techniques et métadonnées descriptives et métadonnées de gestion (API)

: : Etant donné les tests effectués sur le tenant 0 Et un fichier SIP nommé data/SIP_OK/ZIP/OK-RULES_TEST.zip Et je télécharge le SIP Quand j'utilise le fichier de requête suivant data/queries/select_multicriteres_md.json Et je recherche les unités archivistiques Alors les metadonnées sont | Title | titre20999999 | | StartDate | 2012-06-20T18 :58 :18 | | EndDate | 2014-12-07T09 :52 :56 |

Scénario Recherche avancée d'archives – recherche d'archives dans un tenant sur la base de critères correspondant à des archives conservées dans un autre tenant (manuel)

: : Etant donné les tests effectués sur le tenant 0 Quand j'utilise le fichier de requête suivant data/queries/select_multicriteres_md.json Et je recherche les unités archivistiques Alors les metadonnées sont | Title | titre20999999 | | StartDate | 2012-06-20T18 :58 :18 | | EndDate | 2014-12-07T09 :52 :56 | Mais les tests effectués sur le tenant 1 Et je recherche les unités archivistiques Alors le nombre de résultat est 0 “”” { “$roots” : [], “$query” : [{ “$and” : [{ “$eq” : { “#management.AccessRule.Rules.Rule” : “ACC-00002” } }, { “$match” : { “Title” : “titre20999999” } }], “$depth” : 20 }], “$filter” : { “$orderby” : { “TransactedDate” : 1 } }, “$projection” : { } } “”“

Fonctionnalité Modification interdite via API

Scénario KO_UPDATE_UNIT__ID : Vérifier la non modification de _id

: : Etant donné les tests effectués sur le tenant 0 Quand je modifie l'unité archivistique avec la requete “”” {“$query” : [],”$filter” : {},”$action” : [{“$set” : { “_id” : “toto_id” }}]} “”” Et le statut de la requete est Bad Request

Fonctionnalité Affichage des métadonnées de l'objet physique

Scénario CAS OK = import SIP OK et métadonnées de l'objet physique OK

: : Etant donné les tests effectués sur le tenant 0 Et un fichier SIP nommé data/SIP_OK/ZIP/OK_ArchivesPhysiques.zip Quand je télécharge le SIP Alors le statut final du journal des opérations est OK Quand j'utilise la requête suivante “”” { “$roots” : [], “$query” : [{“$and” :[{“$eq” :{“Title” :”Sed blandit mi dolor”}},{“$in” :{“#operations” :[”Operation-Id”]}}], “$depth” : 0}], “$projection” : { “$fields” : { “TransactedDate” : 1, “#id” : 1, “Title” : 1, “#object” : 1, “DescriptionLevel” : 1, “EndDate” : 1, “StartDate” : 1 }}} “”” Et je recherche les groupes d'objets des unités archivistiques Alors les metadonnées sont | #qualifiers.PhysicalMaster.versions.0.DataObjectVersion | PhysicalMaster_1 | | #qualifiers.PhysicalMaster.versions.0.PhysicalDimensions.Height.value

| 21 | | #qualifiers.PhysicalMaster.versions.0.PhysicalDimensions.Height.unit | centimetre | | #qualifiers.PhysicalMaster.versions.0.PhysicalDimensions.Length.value | 29.7 | | #qualifiers.PhysicalMaster.versions.0.PhysicalDimensions.Length.unit | centimetre | | #qualifiers.PhysicalMaster.versions.0.PhysicalDimensions.Weight.value | 1 | | #qualifiers.PhysicalMaster.versions.0.PhysicalDimensions.Weight.unit | kilogram | | #qualifiers.BinaryMaster.versions.0.DataObjectVersion | BinaryMaster_1 | | #qualifiers.BinaryMaster.versions.0.FileInfo.Filename | Filename0 | | #qualifiers.BinaryMaster.versions.0.FormatIdentification.FormatId | fmt/18 |

5.2 Collection FileRules

Fonctionnalité Recherche de règle de gestion

Scénario Vérification et import des règles OK, recherche par id OK

: : Quand je vérifie le fichier nommé data/rules/jeu_donnees_OK_regles_CSV_regles.csv pour le référentiel RULES | Quand j'utilise le fichier de requête suivant data/queries/select_rule_by_id.json Et je recherche les données dans le référentiel RULES Alors le nombre de résultat est 1 Et les metadonnées sont | RuleId | APP-00001 “”“” { “$query” : { “$eq” : { “RuleId” : “APP-00001” } }, “$projection” : { “$fields” : { “#id” : 1, “RuleId” : 1, “Name” : 1 } }, “$filter” : {} } “”“”

5.3 Collection AccessAccessionRegister

Fonctionnalité Recherche dans les registres de fond

Contexte Avant de lancer cette suite de test, je présuppose que les règles de gestions et de formats sont chargés.

Scénario Upload d'un SIP et vérification du contenu dans le registre de fonds

: : Etant donné les tests effectués sur le tenant 0 Et un fichier SIP nommé data/SIP_OK/ZIP/OK_ARBO-COMPLEXE.zip Quand je télécharge le SIP

Et j'utilise le fichier de requête suivant data/queries/select_accession_register_by_id.json

Et je recherche les détails des registres de fond pour le service producteur Vitam Alors les metadonnées sont

OriginatingAgency | Vitam |

SubmissionAgency | Vitam |

ArchivalAgreement | ArchivalAgreement0 |

{

“$query” : {

“$eq” : { “#id” : “Operation-Id”

}

}, “$projection” : {}, “$filter” : {}