



VITAM - Scénarios de test

Version 0.20.0

VITAM

juil. 21, 2017

1	Objectif du document	1
2	Outils de tests	3
3	Tests Manuels	5
3.1	Cahier de tests manuels	5
3.2	Postman	5
3.3	Requêtes DSL	6
4	Tests Automatisés	9
4.1	Cucumber	9
4.2	Tests de stockage	10
4.3	Séquencement de tests	10
5	Scenario pour l'ingest	11
5.1	Liste des scenarii	11

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 test automatisé.

Outils de tests

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

- Les tests manuels disposent d'une grande amplitude d'action
- 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 a vocation à présenter la manière d'interroger le DSL est présent dans ce même document `doc/fr/configuration-tnr/configuration.rst`
- Le tableau du cahier des tests manuels se trouvent dans l'outil Jalios

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 et les objets par collection (au sens MongoDB) ou pour la totalité des collections (sauf le référentiel des formats) présent sur cette IHM. Il n'est cependant pas possible actuellement de purger les référentiels non affichés sur l'IHM recette (contexte ou profil, par exemple).

Tests Manuels

Les tests manuels peuvent être effectués :

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

3.1 Cahier de tests manuels

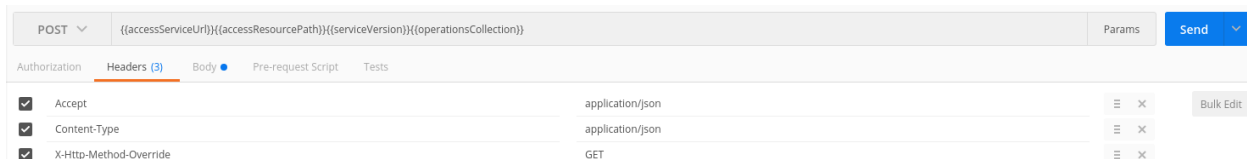
Le cahier de test se présente sous forme de tableur. Il répertorie méticuleusement chaque cas de test possible.

Le tableau contient :

- Le titre explicite du cas de test
- L'itération à laquelle le test se raccroche
- La nature du test (TNR ou Manuel)
- Numéro du bug associé, s'il existe
- La liste des User Stories qui traitent ce cas de test
- Nom de l'activité, nom associé 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
- Le ou les jeux de tests associés

3.2 Postman

Postman est un plugin disponible via Google Chrome et peut être utilisé pour tester les services API. Il s'agit en réalité d'un client HTTP puissant pour tester les services Web. Suite à l'installation d'un certificat, propre à la solution logicielle Vitam, des requêtes DSL peuvent être lancées en GET ou POST.



```

1 {
2   "$query": {},
3   "$filter": {},
4   "$projection": {}
5 }
    
```

Les résultats seront ensuite retournés sous format JSON.

```

1 {
2   "$hits": {
3     "total": 213,
4     "offset": 0,
5     "limit": 1,
6     "size": 213
7   },
8   "$results": [
9     {
10    "_id": "aeaaaaaaaaam7mxablqgakzw3crdeqaaaq",
11    "evId": "aeaaaaaaaaam7mxablqgakzw3crdeqaaaq",
12    "evType": "STP_REFERENTIAL_FORMAT_IMPORT",
13    "evDateTime": "2017-01-19T12:48:05.541",
14    "evDetData": null,
15    "evIdProc": "aeaaaaaaaaam7mxablqgakzw3crdeqaaaq",
16    "evTypeProc": "MASTERDATA",
17    "outcome": "STARTED",
18    "outDetail": null,
19    "outMessg": "Lancement de l'import du référentiel de format",
20    "agId": "{\"Name\":\"vitam-iaas-app-02\",\"Role\":\"functional-administration\",\"PlatformId\":425367}",
21    "agIdApp": null,
22    "agIdAppSession": null,
23    "evIdReq": "aeaaaaaaaaam7mxablqgakzw3crdeqaaaq",
24    "agIdSubm": null,
25    "agIdOrig": null,
26    "obid": null,
27    "obIdReq": null,
28    "obIdIn": null,
29    "events": [
30      {
31        "evId": "aeaaaaaaaaam7mxablqgakzw3crepaaaaq",
32        "evType": "STP_REFERENTIAL_FORMAT_IMPORT",
33        "evDateTime": "2017-01-19T12:48:08.158",
34        "evDetData": null,
35        "evIdProc": "aeaaaaaaaaam7mxablqgakzw3crdeqaaaq",
36        "evTypeProc": "MASTERDATA",
37        "outcome": "OK",
38        "outDetail": null,
39        "outMessg": "Succès de l'import du référentiel de format version 88 du fichier de signature PRONOM (DROID_SignatureFile)",
40        "agId": "{\"Name\":\"vitam-iaas-app-02\",\"Role\":\"functional-administration\",\"PlatformId\":425367}",
    
```

Pour les tests manuels ou exploratoires, Postman est un bon choix pour tester une API. Avec Postman, presque toutes les données d’API Web modernes peuvent être extraites.

Les 2 fonctionnalités pertinentes à retenir : - Ecrirure des tests booléens dans Postman Interface - Création de collections d’appels REST et enregistrement de chaque appel dans le cadre d’une collection à exécuter ultérieurement.

Contrairement à CURL, Postman n’est pas un outil en ligne de commande, ce qui rend cet outil sans tracas dans la fenêtre de ligne de commande. Pour lancer les collections de postman en ligne de commande, on peut installer <https://www.npmjs.com/package/newman>

3.3 Requêtes DSL

Il est possible de lancer des requêtes DSL via l’IHM de recette depuis le menu “Requêtes DSL”, sans besoin de certificat. Cela permet de tester de manière simple et rapide des requêtes DSL.

Il s’agit d’un formulaire permettant de gérer plusieurs variables, telles que le tenant, le contrat d’accès, la collection, l’action testée et un identifiant. La requête est ensuite placée dans un champ texte.

Il est possible de vérifier sa validité avant de la lancer. Les résultats sont ensuite retournés sous format JSON.

REQUÊTES DSL

Tenant	Contrat	Collection	Action	Identifiant
<input type="text" value="1"/>	<input type="text" value="ContratTNR"/>	<input type="text" value="Unit"/>	<input type="text" value="Rechercher"/>	<input type="text"/>

Requête DSL (format JSON)

Vérifier JSON Envoyer requête

Requête DSL (format JSON)

```

    }
  ],
  "$depth": 20
}
],
"$filter": {
  "$orderby": {
    "TransactedDate": 1
  }
},
"$projection": {
  "$fields": {
    "TransactedDate": 1,
    "#id": 1,
    "#unitType": 1,
    "Title": 1,
    "#object": 1
  }
}
}
}

```

Réponse

```

"#object": ""
},
{
  "Title": "SIP_BM_TC",
  "_unitType": "INGEST",
  "#id": "aeaqaahhhy4xrabeigak4pr4lfvaaaaba",
  "#tenant": 0,
  "#object": ""
},
{
  "Title": "ArchiveUnit Contenant un BinaryMaster_1 et un
  TextContent_1",
  "TransactedDate": "2016-10-18T21:03:30",
  "_unitType": "INGEST",
  "#id": "aeaqaahhhy4xrabeigak4pr4jtzaaaaq",
  "#tenant": 0,
  "#object": "aebaaaaahhy4xrabeigak4pr4jtviaaaba"
},
{
  "Title": "ArchiveUnit Contenant un BinaryMaster et un

```

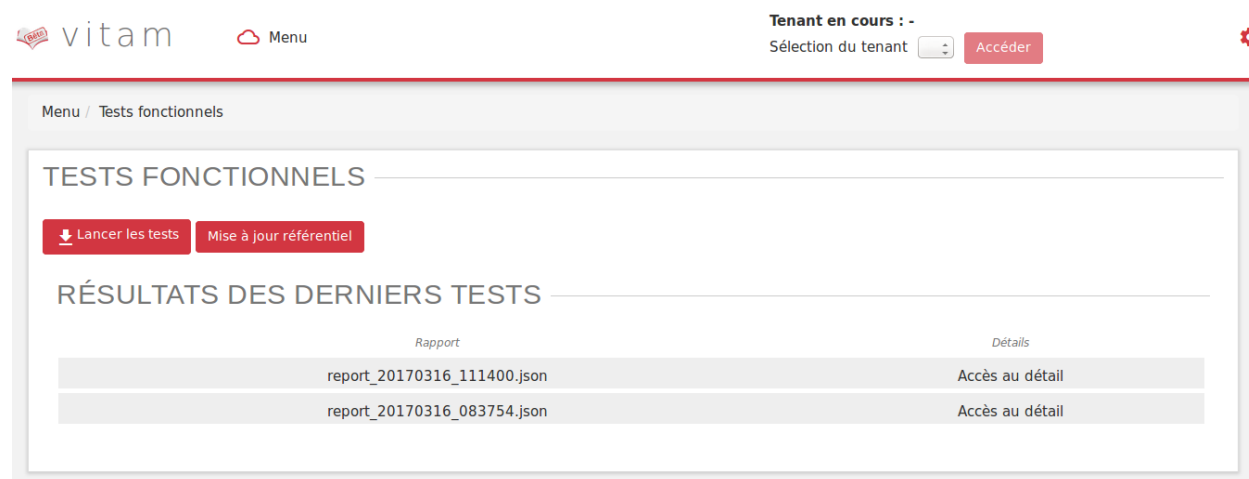

Tests Automatisés

4.1 Cucumber

Cucumber est un outil de tests fonctionnels, il est accessible via l'IHM de recette dans le menu "Tests fonctionnels". Ces tests sont effectués via des ordres écrit 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



Tenant en cours : -
Sélection du tenant [Accéder](#)

Menu / Tests fonctionnels

TESTS FONCTIONNELS

[Lancer les tests](#) [Mise à jour référentiel](#)

RÉSULTATS DES DERNIERS TESTS

Rapport	Détails
report_20170316_111400.json	Accès au détail
report_20170316_083754.json	Accès au détail

RÉSULTATS DES DERNIERS TESTS

Résumé

Nb Tests	Succès	Echecs
3	0	3

Détails

Feature	ID Opération	Description	Errors
✖ Calcul des règles de gestion		Recherche une archive unit avec les règles héritées en cas de la prévention d'héritage (PreventInheritance)	java.nio.file.NoSuchFileException: /vitam/data/ihm-recette/test-data/system/data/SIP_OK/ZIP/1066_CA6_corrige.zip at sun.nio.fs.UnixException.translateToIOException(UnixException.java:86) at sun.nio.fs.UnixException.rethrowAsIOException(UnixException.java:102) at sun.nio.fs.UnixException.rethrowAsIOException(UnixException.java:107) at sun.nio.fs.UnixFileSystemProvider.newByteChannel(UnixFileSystemProvider.java:214) at java.nio.file.Files.newByteChannel(Files.java:361) at java.nio.file.Files.newByteChannel(Files.java:407) at java.nio.file.spi.FileSystemProvider.newInputStream(FileSystemProvider.java:384) at java.nio.file.Files.newInputStream(Files.java:152) at fr.gouv.vitam.functionaltest.cucumber.step.IngestStep.upload_this_sip(IngestStep.java:90) at ✖.Quand je télécharge un SIP
👤 uploader des fichier SIP	aedqaaaaacgbcacaajvsak22er2kzyaaaaq		SIP au mauvais format
👤 uploader des fichier SIP	aedqaaaaacgbcacaajvsak22er3dnyaaaaq		Test virus

4.2 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 stratégie de stockage
- La liste des stratégies où est stocké l'objet
- La présence de l'objet dans ces stratégies

4.3 Séquencement de tests

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

Scenario pour l'ingest

Cette partie décrit les scenario de test correspondant au processus d'ingest.

5.1 Liste des scenarii

Ci-dessous est représentée la liste des différents scenarios de test exécutés dans le cadre de l'automatisation des tests.

Tableau 5.1: L

Nom du Sip	Etat	Code	Nom du test
SIP_OK/ZIP/OK_SIP_2_GO.zip	OK	200	Entrée en succès
SIP_KO/ZIP/KO_VIRUS_code2.zip	KO	200	Échec du processus du contrôle sanitaire du SIP : fichier
SIP_KO/ZIP/KO_BORD_mauvais_format.zip	KO	200	bordereau de versement au mauvais format
SIP_OK/ZIP/OK_SIP_2_GO.zip	OK	200	Succès du processus de vérification du bordereau de vers
SIP_KO/ZIP/KO_BORD_non_conforme_seda.zip	KO	200	Échec du processus de vérification du bordereau de vers
SIP_OK/ZIP/OK_SIP_2_GO.zip	OK	200	Succès du processus de vérification du bordereau de vers
SIP_KO/ZIP/KO_SIP_usages_errores.zip	KO	200	liste des BinaryDataObject et PhysicalDataObject dont
SIP_OK/ZIP/OK_SIP_2_GO.zip	OK	200	Succès du processus de vérification des usages des grou
SIP_OK/ZIP/OK_SIP_2_GO.zip	OK	200	Succès du processus vérification du nombre d'Objets
SIP_KO/ZIP/KO_OBJT_nombresup_SEDA.zip	KO	200	Échec du processus de vérification du nombre d'Objets
SIP_OK/ZIP/OK_SIP_2_GO.zip	OK	200	Succès du processus de vérification de l'empreinte
SIP_KO/ZIP/KO_BORD empreinteKO.zip	KO	200	Échec du processus de vérification de l'empreinte : liste
SIP_KO/ZIP/KO_OBJT_orphelins.zip	KO	200	Échec du processus de contrôle métier et extraction du S
SIP_OK/ZIP/OK_SIP_2_GO.zip	OK	200	Succès du processus de contrôle métier et extraction du
SIP_OK/ZIP/OK_ARBO_rateau.zip	OK	200	Succès du processus de création du Journal de Cycle de
SIP_OK/ZIP/Format_ID_Different.zip	WARNING	206	Identification des formats, FormatId différents
SIP_OK/ZIP/OK_FORMT_PUID_incoherent.zip	WARNING	206	Avertissement lors du processus de vérification des form
SIP_OK/ZIP/OK_ARBO_rateau.zip	OK	200	Succès du processus de contrôle globale de l'entrée
SIP_KO/ZIP/KO_BORD_mauvais_format.zip	KO	200	Échec du processus de contrôle globale de l'entrée
SIP_OK/ZIP/OK_SIP_2_GO.zip	OK	200	Succès du processus de contrôle et traitements des Unité
SIP_OK/ZIP/OK_SIP_2_GO.zip	OK	200	Succès du processus de vérification préalable à la prise
SIP_OK/ZIP/OK_SIP_2_GO.zip	OK	200	Succès du processus de vérification de la disponibilité d
SIP_OK/ZIP/OK_SIP_test_différentes_langues.zip	OK	200	Succès de la sécurisation des métadonnées des Unités A
SIP_OK/ZIP/OK_SIP_test_différentes_langues.zip	OK	200	Succès du processus d'indexation des métadonnées des
SIP_OK/ZIP/OK_SIP_test_différentes_langues.zip	OK	200	Succès du processus d'enregistrement du journal de cyc
SIP_OK/ZIP/OK_ARBO_rateau.zip	OK	200	Succès du processus de rangement des Unités Archiviste

Tableau 5.1 –

Nom du Sip	Etat	Code	Nom du test
SIP_OK/ZIP/OK_ARBO_rateau.zip	OK	200	Succès du processus de rangement des Objets et groupe
SIP_OK/ZIP/OK_SIP_2_GO.zip	OK	200	Succès du processus d'indexation des métadonnées des
SIP_OK/ZIP/OK_SIP_2_GO.zip	OK	200	Succès du processus d'enregistrement du journal de cyc
SIP_OK/ZIP/OK_SIP_2_GO.zip	OK	200	Succès du processus de rangement des Objets et groupe
SIP_OK/ZIP/OK_SIP_2_GO.zip	OK	200	Succès du processus de finalisation de l'entrée et de not
SIP_OK/ZIP/OK_SIP_2_GO.zip	OK	200	Succès du processus de notification à l'opérateur de ver
SIP_OK/ZIP/OK_SIP_2_GO.zip	OK	200	Succès du processus d'alimentation du Registre des For