VITAM - Scénarios de test

Version 1.10.0-1

VITAM

nov. 19, 2018

Table des matières

CHAPITRE 1

Objectif du document

Ce document présente les différentes méthodes et outils permettant de 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.

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.

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 archivistiques et groupes d'objets et les contrats par collection (au sens MongoDB) ou pour la totalité des collections présentent dans cette IHM.

CHAPITRE 2

Tests Manuels

Les tests manuels peuvent être effectués :

- A l'aide du cahier de tests manuels.
- Via l'IHM recette, au travers de requêtes DSL

2.1 Cahier de tests manuels

Le cahier de test manuel se présente sous forme de tableur. Il répertorie tous les cas de test possibles, regroupés par onglets par grand domaine fonctionnel.

Ce document est disponible dans la documentation de la solution logicielle Vitam. Pour les partenaires du programme Vitam, une copie se trouve également dans l'outil Jalios, dans l'espace livraison.

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 à quelle interface le test est dédié
- Le ou les jeux de tests associés

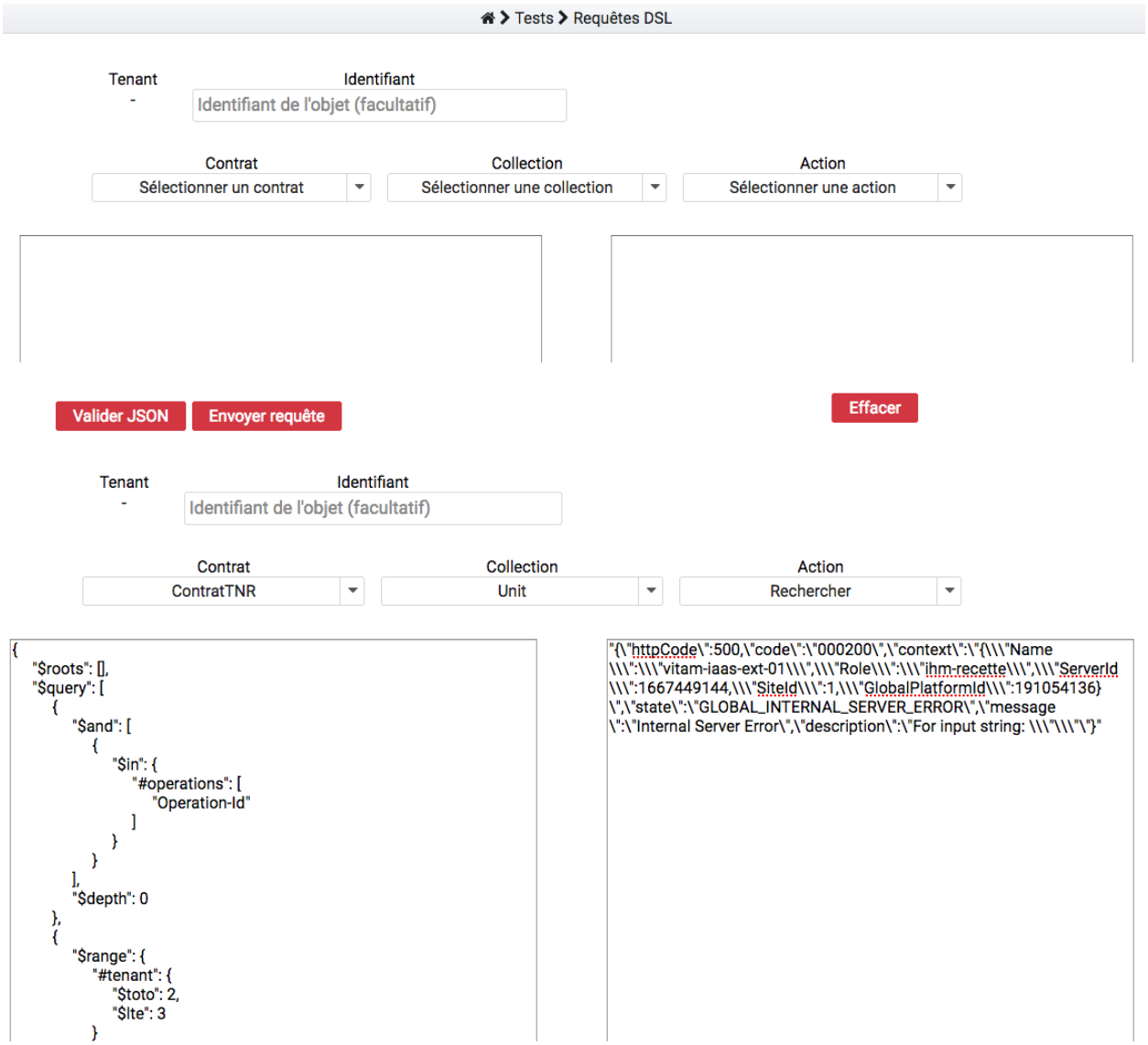
2.2 Requêtes DSL

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 tenant doit être sélectionné au préalable au niveau du menu.

Un formulaire permet de gérer plusieurs variables. Au niveau du formulaire, il faut choisir :

- Un contrat d'accès sur lequel lancer le test
- La collection relative à la requête
- L'action à tester (une recherche ou une mise à jour)
- Un identifiant (obligatoire ou non selon la requête effectuée)

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 3

Tests Automatisés

3.1 Principes généraux

3.1.1 Tests de non régression

Les Tests de Non Régression (TNR) ont pour objectif de tester la continuité des fonctionnalités de Vitam. L'ajout de nouvelles fonctionnalités pouvant entraîner des bugs ou anomalies (régressions) sur des fonctionnalités existantes. L'outil de test de non régression va permettre de tester automatiquement le périmètre fonctionnel pré-existant afin de s'assurer de son bon fonctionnement dans le temps. Les TNR peuvent aussi être utilisés comme indicateur de bonne santé d'une plateforme.

L'ajout d'une nouvelle fonctionnalité dans la solution logicielle Vitam et parfois la correction d'un bug s'accompagne d'un ou plusieurs TNR.

Idéalement, les développeurs doivent lancer les TNR avant d'effectuer une Merge Request visant à intégrer une nouvelle fonctionnalité, afin de valider que le nouveau code introduit ne provoque pas de régressions dans le reste de la solution logicielle Vitam.

Suite à une nouvelle installation du produit, le lancement des TNR permet également de vérifier le bon déploiement de la solution logicielle.

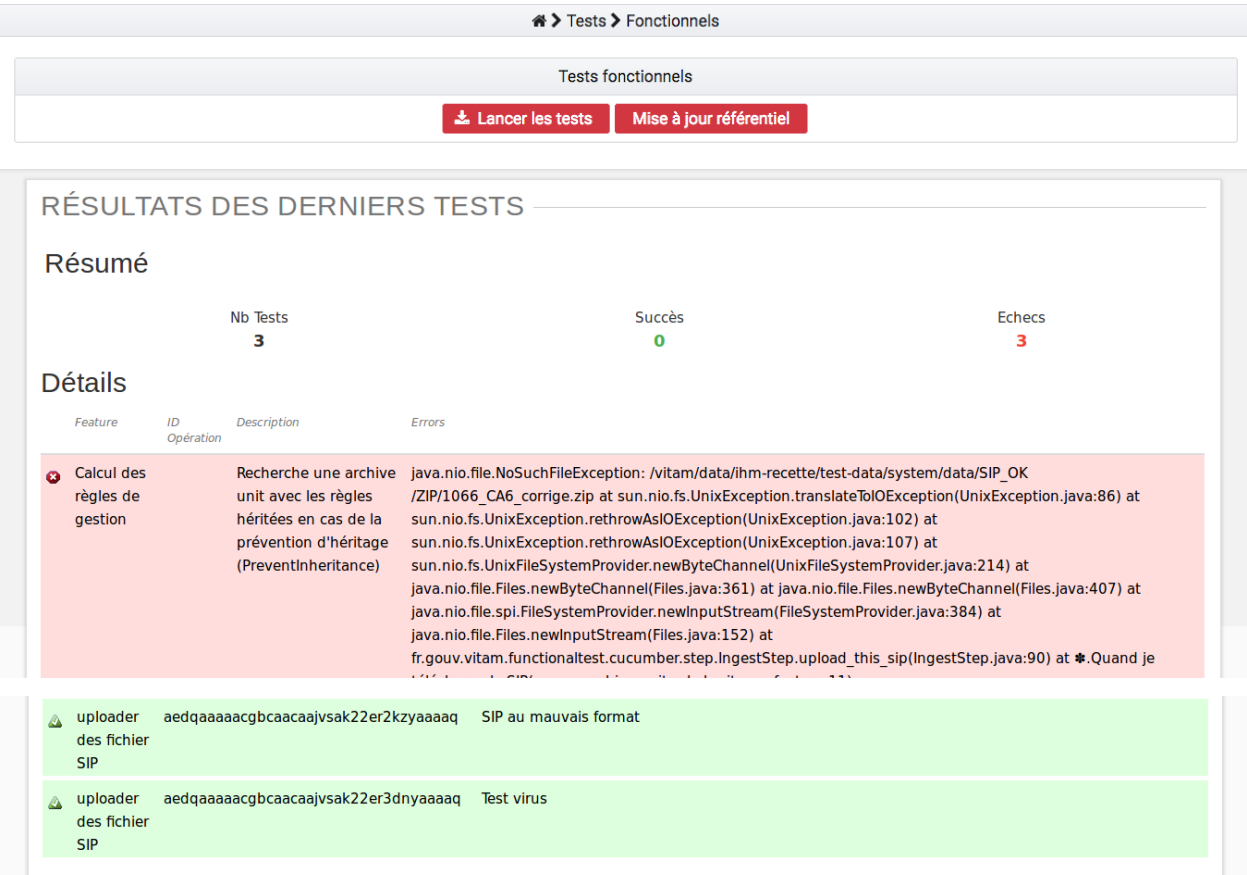
3.1.2 Tests fonctionnels

Cucumber

Cucumber est un outil de tests fonctionnels, il est accessible via l'IHM recette dans le menu « Tests / Tests fonctionnels ». Ces tests sont effectués via des ordres écrits avec des phrases simples offrant 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



3.1.3 Behavior-Driven Development (BDD)

Le BDD est une méthode de collaboration s'appuyant sur un langage de programmation naturel permettant aux intervenants non techniques de décrire des scénarios de fonctionnement.

Les mots de ce langage permettent de mobiliser des actions techniques qui sont, elles, réalisées par les développeurs.

Le BDD est utilisé pour la réalisation des TNR, ce qui permet à tout intervenant du projet de pouvoir en réaliser.

Le framework de test utilisé dans le cadre de Vitam est Cucumber (https://cucumber.io/) qui utilise le langage Gherkin (https://github.com/cucumber/cucumber/wiki/Gherkin)

3.2 Pré-Requis

3.2.1 Dépot vitam-itest

La liste des TNR existants ainsi que tous leurs jeux de données associés sont déposés dans le dépôt git vitam-itest et également dans les ressources publiées à chaque release.

Il est donc nécessaire de le cloner avant toute chose.

3.2.2 Git LFS

Afin de permettre la gestion de fichiers volumineux dans git, il est nécessaire d'installer l'extension Git-LFS (https://git-lfs.github.com/).

Une fois git lfs installé, il est nécessaire de l'activer pour le dépôt vitam-itest sur votre environnement. Pour réaliser cette opération, il faut se placer à la racine du dépôt et exécuter la commande :

```
git lfs install
```

3.3 Méthodologie de test

3.3.1 Séquencement

Les tests sont regroupés par lot intellectuellement cohérent dans des fichiers .feature. Chaque fichier .feature contient au moins un scénario de test. Lorsqu'un fichier feature est lancé, alors tous les scénarios qu'il spécifie sont exécutés séquentiellement. Lorsqu'un scénario est en échec alors que son exécution n'est pas terminée, celle-ci s'interrompt et le scénario suivant est lancé.

Les fichiers feature sont lancés dans l'ordre alphabétique et sont indépendants les uns des autres. Il est donc possible d'exécuter une sélection de tests, sans devoir se soucier de dépendances inter-fichiers.

Il y a une exception à ce principe : le fichier nommé « _init.feature » est un scénario qui met en place l'environnement de test en important les ressources nécessaires (référentiel des règles de gestions, des contrats, des services agents. . .) à la bonne exécution des tests suivants.

3.3.2 Lancement complet des TNR

Il est possible de lancer tous les TNR via l'IHM recette en allant dans le menu Tests > Tests fonctionnels puis de cliquer sur le bouton « Lancer les tests ». Lors de ce processus, les bases sont vidées avant le lancement de chaque campagne de test. Comme vu précédemment, elles sont immédiatement réinitialisées avec des données de test, suite à l'exécution en premier du fichier _init.feature.

CHAPITRE 4

Ecriture des TNR

4.1 Structure des répertoires

Le répertoire du dépot vitam-itest est strucutré de la façon suivante :

```
vitam-itests
                |------- data
```

Dossier vitam-itests : contient les fichiers de configurations des tests fonctionnels.

Dossier data : contient les éventuels jeux de données nécessaires à l'exécution des tests.

4.2 Fichiers de Configuration

4.2.1 Nommage des fichiers

Un fichier regroupe tous les tests à effectuer sur une fonctionnalité. Il ne peut y avoir deux fonctionnalités dans un fichier de configuration.

On va par exemple réaliser :

- un fichier pour les tests sur l'Ingest
- un fichier pour les test sur l'accès aux unités archivistiques

Les noms des fichiers sont composés de la façon suivante :

```
EndPoint-Fonctionnalité.feature
```

Par exemple :

```
access-archive-unit.feature
admin-logbook-traceability.feature
```

4.2.2 Informations transverses

Les fichiers de configuration doivent contenir les informations suivantes qui s'appliqueront ensuite à l'ensemble des scénarios du fichier :

 # language : information obligatoire. Correspond à la langue utilisée pour les descriptions. Par exemple :

```
language : fr.
```

Annotation : information optionnelle. L'annotation permet par la suite de lancer uniquement un fichier de configuration en ligne de commande en utilisant son annotation en paramètre. Par exemple :

```
@AccessArchiveUnit
```

Fonctionnalité : information obligatoire. La fonctionnalité est une description qui permet d'identifier le périmètre testé. Il est notamment repris dans les rapports réalisés à la fin d'une campagne de test. Par exemple :

```
Fonctionnalité: Recherche une archive unit existante
```

Contexte : information optionnelle. Les informations contenues dans contexte sont des actions qui vont s'exécuter pour chacun des scénarios. A ce titre, elles s'écrivent comme les actions d'un scénario. Le constexte doit être indenté de 1 par rapport aux autres éléments. Par exemple :

```
Contexte:
Etant donné les tests effectués sur le tenant 0
```

4.3 Ecriture d'un scénario

4.3.1 Structure d'un scénario

Un scénario correspond à un test. Son nom doit être défini de la façon suivante :

```
Scénario: Description du scénario
```

Il doit être sur la même indentation que le contexte, soit 1 par rapport à la fonctionnalité, à l'annotation et au langage.

Un scénario est constitué d'une succession d'actions, chacune décrite sur une ligne.

Les actions sont composées des trois informations suivantes :

- Contexte
- Fonction
- Paramètre (pas toujours obligatoire)

Actions d'étapes : permet d'introduire l'action, de l'insérer par rapport à l'action précédente. La liste des contextes disponibles se trouve en annexe.

Fonction : mobilise, via un langage naturel, une fonction de Vitam. La liste des fonctions disponibles se trouve en annexe.

Paramètre : certaines fonctions ont besoin d'être suivies d'un paramètre. Ils sont listés dans le tableau des fonctionnalités disponibles en annexe.

Les actions doivent être indentées de 1 par rapport aux scénarios.

Exemple d'un scénario constitué de trois actions :

```
Scénario: SIP au mauvais format
  Etant donné un fichier SIP nommé data/SIP_KO/ZIP/KO_SIP_Mauvais_Format.pdf
  Quand je télécharge le SIP
  Alors le statut final du journal des opérations est KO
```

4.3.2 Insérer une requète DSL

Certaines fonctions nécessitent l'entrée de requêtes DSL en paramètre. Celles-ci doivent être insérées entre guillemets (» «), après un retour à la ligne à la suite de la fonction.

Voici un exemple d'une action suivie d'une requête DSL :

```
Et j'utilise la requête suivante
        """
        { "$roots": [],
                "$query": [
                { "$and": [
                        { "$gte": {
                        "StartDate": "1914-01-01T23:00:00.000Z"
                                } }, {"$lte": {
                                "EndDate": "1918-12-31T22:59:59.000Z"
                                } } ],
                        "$depth": 20}],
                        "$filter": {"$orderby": { "TransactedDate": 1 }
                        }, "$projection": {
                        "$fields": {"TransactedDate": 1, "#id": 1, "Title": 1, "
↪#object": 1, "DescriptionLevel": 1, "EndDate": 1, "StartDate": 1}}}
        """
```

4.3.3 Insérer un tableau

Certaines fonctions attendent un tableau en paramètre. Les lignes des tableaux doivent simplement être séparées par des « pipes » (|).

Voici un exemple de fonction prenant un tableau en paramètre.

```
  Alors les metadonnées sont
| Title            | Liste des armements |
| DescriptionLevel | Item                |
| StartDate        | 1917-01-01          |
| EndDate          | 1918-01-01          |
```

4.4 Annexes

4.4.1 Liste des actions d'étapes disponibles

Les types d'actions sont les suivants :

- une situation initial (les acquis) : **Etant donné**
- un événement survient : **Quand** (peut être suivi de **Et** et/ou **Mais**)
- on s'assure de l'obtention de certains résultats : **Alors** (peut être suivi de **Et** et/ou **Mais**)

Action
Etant donné
Quand
Alors
Mais
Et

4.4.2 Liste des fonctions disponibles

Fonctionnalité	Doit être suivi par
les tests effectués sur le tenant (*)	un tenant
les données du jeu de test du SIP nommé (.*)	un fichier
un fichier SIP nommé (.*)	un fichier
je télécharge le SIP	une autre action
je recherche le journal des opérations	une autre action
je télécharge son fichier ATR	une autre action
je recherche le JCV de l'unité archivistique dont le titre est (.*)	un titre d'unité archivistique
je recherche le JCV du groupe d'objet de l'unité archivistique dont le titre est (.*)	un titre d'unité archivistique
le statut final du journal des opérations est (.*)	un statut
le[s] ? statut[s] ? (? :de l'événement|des événements) (.*) (? :est|sont) (.*)	un ou plusieur evType et un Statut
l'outcome détail de l'événement (.*) est (.*)	un outcome detail et une valeur
l'état final du fichier ATR est (.*)	un statut
le fichier ATR contient (.*) balise[s] de type (.*)	un nombre et un type de balise
le fichier ATR contient les valeurs (.*)	une ou plusieurs valeurs séparées par des virgules
le fichier ATR contient la chaîne de caractères (.*)	un texte ou une simple chaîne de caractères
j'utilise la requête suivante	une requête
j'utilise le fichier de requête suivant (.*)	un fichier
j'utilise dans la requête le GUID de l'unité archivistique pour le titre (.*)	un titre d'unité archivistique
j'utilise dans la requête le paramètre (.*) avec la valeur (.*)	un nom de paramêtre et une valeur de remplacem
je recherche les unités archivistiques	une autre action
je recherche les groupes d'objets des unités archivistiques	une autre action
je recherche les groupes d'objets de l'unité archivistique dont le titre est (.*)	un titre d'unité archivistique
le nombre de résultat est (.*)	un nombre
les metadonnées sont (.*)	un tableau
les metadonnées pour le résultat (.*)	un nombre et un tableau
je recherche les registres de fonds (.*)	une autre action
le nombre de registres de fonds est (.*)	un nombre
les metadonnées pour le registre de fond sont	un tableau
je recherche les détails du registre des fonds pour le service producteur (.*)	un identifiant de service producteur
le nombre de détails du registre des fonds est (.*)	un nombre
les metadonnées pour le détail du registre des fonds sont	un tableau

CHAPITRE 5

Guide d'écriture des tests Cucumber

Voici des exemples sur les types de test qui peuvent être écrits avec l'outil Cucumber paramétré sur le projet Vitam.

5.1 Guide technique

Les exemples suivants présentent l'exhaustivité des phrases définies par fonctionnalité.

5.1.1 Fonctionnalité : ingest

Scénario : Envoi d'une archive et vérification du journal de l'opération, de l'ATR, des journaux de cycle de vie d'un Unit et d'un GOT

Contexte Avant de lancer ce scénario, je présuppose que les contrats d'entrée, les contrats d'accès, les référentiels des règles de gestions et des formats sont chargés.

```
Scénario: Guide ingest
# Execution d'un ingest
  Etant donné un fichier SIP nommé data/SIP_GUIDE_INGEST_OK.zip
  Quand je télécharge le SIP
# Vérification du journal des opérations
  Et je recherche le journal des opérations
  Alors le statut final du journal des opérations est KO
  Et les statuts des événements CHECK_DIGEST, STP_OG_CHECK_AND_TRANSFORME sont KO
  Et l'outcome détail de l'événement CHECK_DIGEST est CHECK_DIGEST.KO
  Et l'outcome détail de l'événement STP_OG_CHECK_AND_TRANSFORME est STP_OG_CHECK_AND_
↪TRANSFORME.KO
# Vérification de l'ATR
  Quand je télécharge son fichier ATR
  Alors l'état final du fichier ATR est KO
  Et le fichier ATR contient 1 balise de type Date
  Et le fichier ATR contient les valeurs STP_OG_CHECK_AND_TRANSFORME, CHECK_DIGEST,␣
↪LFC.CHECK_DIGEST, LFC.CHECK_DIGEST.CALC_CHECK
```

(suite sur la page suivante)

(suite de la page précédente)

```
  Et le fichier ATR contient la  chaîne de caractères
"""
<BinaryDataObject id="ID018">
"""
  Et le fichier ATR contient la  chaîne de caractères
"""
<ArchiveUnit id="ID019">
"""
# Vérification du JCV d'un des Units
  Quand je recherche le JCV de l'unité archivistique dont le titre est Fichier 2␣
↪nouveau jeu de test
  Alors les statuts des événements LFC.UNITS_RULES_COMPUTE, LFC.UNIT_METADATA_
↪INDEXATION, LFC.UNIT_METADATA_STORAGE sont OK
# Vérification du JCV d'un des GOTs
  Quand je recherche le JCV du groupe d'objet de l'unité archivistique dont le titre␣
↪est Historique de la station Gambetta
  Alors les statuts des événements LFC.OG_OBJECTS_FORMAT_CHECK.FILE_FORMAT,LFC.OG_
↪OBJECTS_FORMAT_CHECK est OK
```

5.1.2 Fonctionnalité : recherche simple des métadonnées d'AUs et de GOTs

Scénario Recherche de toutes les unités archivistiques et de groupes d'objets liés à un ingest

Contexte Avant de lancer ce scénario, je présuppose que les contrats d'entrée, les contrats d'accès, les référentiels des règles de gestions et des formats sont chargés.

```
Scénario: Guide recherche simple
# Execution d'un ingest
  Etant donné les tests effectués sur le tenant 0
  Et les données du jeu de test du SIP nommé data/SIP_GUIDE_OK.zip
# Rechercher des AUs
  Quand j'utilise la requête suivante
"""
{ "$roots": [],
  "$query": [{"$in":{"#operations":["Operation-Id"]}}],
  "$filter": {
    "$orderby": { "TransactedDate": 1}
  },
  "$projection": {}
}
"""
  Et je recherche les unités archivistiques
# Vérification des résultats
  Alors le nombre de résultat est 5
  Alors les metadonnées pour le résultat 0
    | inheritedRule.StorageRule.R3.{{unit:AU13}}.path                  | [["{
↪{unit:AU13}}","{{unit:AU14}}"]]
    | inheritedRule.AccessRule.ACC-00002.{{unit:2_Front Populaire}}.path.array[][]  ␣
↪    | [["{{unit:2_Front Populaire}}"]] |      |
    | inheritedRule.AccessRule.ACC-00001.{{unit:AU51}}.EndDate    | "2017-01-01"  ␣
↪                                                 |
    | #management.AccessRule.Inheritance.PreventRulesId.array[]                  ␣
↪            | "ACC-00002" |
    | #management.DisseminationRule.Inheritance.PreventInheritance               ␣
↪| true |
```

(suite sur la page suivante)

(suite de la page précédente)

```
  Quand je recherche les groupes d'objets de l'unité archivistique dont le titre est␣
↪ID8
  Alors les metadonnées sont
    | #qualifiers.1.versions.0.DataObjectVersion                    | BinaryMaster_
↪1      |
    | #qualifiers.1.versions.0.FileInfo.Filename                    | Filename0     ␣
↪       |
    | #qualifiers.1.versions.0.FormatIdentification.FormatId        | fmt/18        ␣
↪       |
```

5.1.3 Fonctionnalité : recherche complexe d'une AU

Scénario Recherche d'une unités archivistique particulière et de son groupe d'objet

Contexte Avant de lancer ce scénario, je présuppose que les contrats d'entrée, les contrats d'accès, les référentiels des règles de gestions et des formats sont chargés.

```
Scénario: Guide recherche avancée
# Execution d'un ingest
  Etant donné les tests effectués sur le tenant 0
  Et les données du jeu de test du SIP nommé data/SIP_GUIDE_OK.zip
# Rechercher complexe avec requête dans un fichier et remplacement de paramètres
  Quand j'utilise le fichier de requête suivant data/queries/query.json
  Et j'utilise dans la requête le GUID de l'unité archivistique pour le titre Archive␣
↪unit ID1
  Et j'utilise dans la requête le paramètre SEDA-ID-UNIT avec la valeur ID1
  Et j'utilise dans la requête le paramètre DEPTH avec la valeur 0
  Et je recherche les unités archivistiques
# Vérification des résultats
  Alors le nombre de résultat est 1
  Alors les metadonnées sont
    | Title            | Archive unit ID0101 |
    | StartDate        | 2012-06-20T18:58:18 |
    | EndDate          | 2014-12-07T09:52:56 |
```

Le fichier *data/queries/query.json* contient :

```
{
  "$roots": ["{{guid}}"],
  "$query": [
    {
      "$and": [
        {
          "$in": { "#operations": ["Operation-Id"] }
        },
        {
          "$eq": { "Title": "Archive unit SEDA-ID-UNIT" }
        }
      ],
      "$depth": DEPTH
    }
  ],
  "$projection": {
    "$fields": {
      "#id": 1,
```

(suite sur la page suivante)

(suite de la page précédente)

```
      "Title": 1
    }
  }
}
```

5.1.4 Fonctionnalité : recherche d'un registre des fonds

Scénario Recherche d'un registre des fonds et de son détail pour une opération d'ingest

Contexte Avant de lancer ce scénario, je présuppose que les contrats d'entrée, les contrats d'accès, les référentiels des règles de gestions et des formats sont chargés.

```
Scénario: Guide registre de fonds
# Execution d'un ingest
  Etant donné un fichier SIP nommé data/SIP_GUIDE_OK.zip
  Quand je télécharge le SIP
  Et je recherche le journal des opérations
  Alors le statut final du journal des opérations est OK
# Rechercher du registre de fonds
  Quand j'utilise la requête suivante
"""
{
  "$query": { "$eq": { "OriginatingAgency": "FRAN_NP_009913" } },
  "$projection": {}
}
"""
  Et je recherche les registres de fond
# Vérification du registre de fonds
  Et le nombre de registres de fond est 1
  Et les metadonnées pour le registre de fond sont
    | OriginatingAgency        | FRAN_NP_009913              |
    | TotalObjects.ingested        | 4 |
    | TotalObjectGroups.ingested       | 4 |
    | TotalUnits.ingested        | 7 |
# Rechercher du détail du registre de fonds pour l'ingest
  Quand j'utilise la requête suivante
"""
{
  "$query": {
    "$and": [ { "$in": { "OperationIds": [ "Operation-Id" ] } } ]
  },
  "$projection": {}
}
"""
  Et je recherche les détails des registres de fond pour le service producteur FRAN_
↪NP_009913
# Vérification du détail du registre de fonds
  Et le nombre de détails du registre de fond est 1
  Et les metadonnées pour le détail du registre de fond sont
    | OriginatingAgency                | FRAN_NP_009913              |
    | TotalObjects.ingested            | 4 |
    | TotalObjectGroups.ingested       | 4 |
    | TotalUnits.ingested              | 7 |
```

5.2 Scénarios fonctionnels

5.2.1 Collection Unit

Fonctionnalité Recherche avancée

Scénario Recherche avancée d'archives – cas OK d'une recherche multicritère croisant métadonnées techniques, 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              |
```

5.2.2 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": {} }
"""
```

5.2.3 Collection AccessAccessionRegister

Fonctionnalité Recherche dans les registres des fonds

Contexte Avant de lancer ce scénario, je présuppose que les contrats d'entrée, les contrats d'accès, les référentiels des règles de gestions et des 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": {}
}
"""
```

5.2.4 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

CHAPITRE 6

Tests curl

6.1 Introduction

cUrl, est l'abréviation de « client URL request library ».

C'est un outil en ligne de commande permettant notamment, dans le cas qui nous intéresse, de simuler des requêtes HTTP.

Il permet entre autres d'exécuter toutes les méthodes offertes par le REST : POST, PUT, GET, DELETE, HEAD.

Cette documentation a pour but de fournir une panoplie de jeux de test curl afin de pouvoir tester au maximum les API offertes par VITAM.

Pour toutes ces requêtes, il conviendra d'adapter l'uri en fonction de l'environnement à tester (eg : remplacer {env.programmevitam.fr}).

6.2 API Externes

6.2.1 Ingest

Ingest d'un fichier se trouvant dans le dossier baseUploadPath Le fichier doit se trouver dans le répertoire configuré dans le fichier de configuration (baseUploadPath) puis, exécuter : `curl -v -X POST -k --key vitam-vitam_1.key --cert vitam-vitam_1.pem '{env.programmevitam.fr}/ingest-external/v1/ingests' -H 'X-Tenant-Id: 0' -H 'X-Access-Contract-Id: ContratTNR' -H 'Content-Type: application/json;charset=UTF-8' -H 'Accept: application/json' -H 'X-Context-Id: DEFAULT_WORKFLOW' -H 'X-ACTION: RESUME' --data-binary '{"path": "SIP_OK_2_0.zip"}'`

6.2.2 Access

6.2.2.1 Units

Mise à jour d'une unité archivistique : `curl -v -X PUT -k --key vitam-vitam_1.key --cert vitam-vitam_1.pem 'https://{env.programmevitam.fr}/access-external/v1/units/aeaqaaaaaahmtusqabz5oalc4p2zu5aaaaaq' -H 'X-Tenant-Id: 0' -H 'X-Access-Contract-Id: ContratTNR' -H 'Content-Type: application/json' -H 'Accept: application/json' --data-binary '{ "$action": [{ "$set": { "Title": "Montparnasse.txt" } }] }' --compressed`

6.2.3 Administration fonctionnelle

Import d'un référentiel d'agents : `curl -v -X POST -k --key vitam-vitam_1.key --cert vitam-vitam_1.pem 'https://{env.programmevitam.fr}/admin-external/v1/agencies' -H 'X-Tenant-Id: 0' -H 'X-Access-Contract-Id: ContratTNR' -H 'Content-Type: application/octet-stream;charset=UTF-8' -H 'Accept: application/json' --data-binary "@agencies.csv" -H 'Transfer-Encoding: chunked' --compressed`

//TODO

6.3 API Internes

6.3.1 Ingest

//TODO

6.3.2 Access

//TODO

6.3.3 Administration fonctionnelle

//TODO

6.4 Notes

6.4.1 Personae

Pour toutes les requêtes curl de cette documentation, il est possible d'y ajouter un header qui sera pris en compte dans les journaux d'opération : X-Personal-Certificate. Pour réaliser une telle prouesse, il s'agit simplement d'ajouter ceci à la commande curl ciblée (modifier le certificat en base 64 avec quelque chose de valable dans l'environnement, cf la collection PersonalCertificate) : `-H 'X-Personal-Certificate:MIIFRjCCAy6gAwIBAgIBAjANBgkqhkiG9w0BAQsFADAtMQswCQYDVQQGEwJGUjEOMAw`
`2AKgtcxscYH/0yG1bDQ3vT2tv0YH4jzdfXfwTVzytqAV1M/CNZlWbcBXqDyZLeYUm5i/`
`Dufndj16j4hw24tBsQT1o92P5qdfPaieZc4jpscGiMmyNYwEKcbqo5wiGVsiD+sU9/`
`JXHT2q1f18JcuwJ5/fqzsADPKXudBvibCSaANf+ZNpRaWZ7y6e/kUDs8yrp4YaXzb331ioOGk4JE9ylv1hY5l8IbbvW`
`VMHrrGlkVjmGBUydJDiRhUAgaqXNpezwWulweQunAelBCU4PjO40J6t2wdLi5+f+b0OLJHJg0N9xdFsKrqsAVpjaYpc`

```
LzkR3cxK+9Pw4xDIEgQ0ZTCvY/6SBnHdAe3tqs4kODs57BZW4DD9ytpT73BKMf7EeZAE3tJd9p40uw/
b41VF9bJvoW1ammZMl4H2OwdJi7+5DAMbBC1X2kMGWRo06IM99q+TKpfrUK+p4b6NfcSdrfr+n28vd8pzp16VDoMCAw
BAIwADAdBgNVHQ4EFgQUx0RltSTqHWXEisK78KQzn2SRpIkwCwYDVR0PBAQDAgSwMBEGCWCGSAGG+EIBAQQEAwIFoDA
D0vh0FqKvlCBTOLJsLfO2hsEL8ude5rVhP4goThIz1OjpnxFP+YmHUOtiQup21VGTaeTWn769/
x6gRx/1eyJyws4ien/w7gBASLEKI7nGYAkeoYeZKWYTlfBgEisLwSsjcQeBeKcnUnuWJauiALPnBntkAnM7PotASA8
l+8f1quMAPoTIlsG+TBFW0s9+LqY8ufE9+8u8S1FynZlsgfIoKl2bKVXWWrZVfJ+S8mh6mH4V3MuhLwljv+/
6HDZCc3FoY5eN/lyWI49Maz5W87bKqNyecYtrBlvML7k5UeOLtgNuUsTBlzFTxMkaQHOSpMyrHZ/
yVPNVfuP3cCKvzMPHFGHzJZK0qvz4zdFdx7YzBq+I6YLvRES9b+DkvdrTOpZI2GjKuP5m13kcUjsFeqJR6rb+o1kJuC
QMC2OjMXMlDqNa8mL5ooGQmYOzHkfq4vdKLG/Fvbpw2DDrwv9jKmw2l6eWLYzuIpvz7sqUHwi30wScXSm/
FCKF9DjzODUpSkBvDiaA==’
```