



# IHM Recette

Date	Version
15/03/2021	9.0 (Release 16 - V4)

## État du document

En projet     Vérifié     Validé

## Maîtrise du document

Responsable	Nom	Entité	Date
Rédaction	Équipe Vitam	Équipe Vitam	09/01/2019
Vérification	MVI	Équipe Vitam	15/12/2020
Validation	AGR	Équipe Vitam	15/03/2021

## Suivi des modifications

Version	Date	Auteur	Modifications
1.0	09/01/2019	MRE	Génération à partir de l'ancien document en RST
1.1	24/01/2019	MAF	Relecture et Corrections
1.2	25/01/2019	MRE	Relecture
2.0	30/01/2019	MRE	Finalisation du document pour publication de la Release 9
2.1	05/04/2019	MAF	Relecture
3.0	24/04/2019	MRE	Finalisation du document pour publication de la Release 10
3.1	25/07/2019	MAF	Mise à jour
4.0	09/09/2019	MAF	Finalisation du document pour publication de la Release 11
4.1	15/11/2019	DAG	Mise à jour du document pour la Release 12
5.0	29/11/2019	AGR	Finalisation du document pour publication de la Release 12
5.1	05/03/2020	MVI	Mise à jour du document pour la Release 13 : <ul style="list-style-type: none"> <li>• chapitre II.5 : ajout de la sous-section « Configuration de la plateforme » ;</li> <li>• chapitre II.6 : ajout de la sous-section « Nettoyage d'ingest corrompu ».</li> </ul>
6.0	20/03/2020	AGR	Finalisation du document pour publication de la Release 13
6.1	28/05/2020	MVI	Relecture
7.0	06/07/2020	AGR	Finalisation du document pour publication de la Release 14
7.1	20/10/2020	MVI	Relecture
8.0	26/10/2020	AGR	Finalisation du document pour publication de la Release 15
8.1	15/12/2020	Equipe	Relecture
9.0	15/03/2021	AGR	Finalisation du document pour publication de la Release 16

## Licence

La solution logicielle VITAM est publiée sous la licence CeCILL 2.1 ; la documentation associée (comprenant le présent document) est publiée sous [Licence Ouverte V2.0](#).

## TABLE DES MATIÈRES

1.Objectif du document.....	6
2.Principes généraux.....	7
2.1.Principes généraux.....	7
2.2.Accès.....	7
2.3.Navigation.....	7
2.4.Fil d’Ariane.....	8
2.5.Titre des onglets.....	8
2.6.Sélection d’un tenant.....	9
3.Administration.....	10
3.1.Administration des collections.....	10
3.1.1.Purger toutes les collections de la solution logicielle Vitam.....	10
3.1.2.Purger les référentiels.....	10
3.1.2.1.Purger les référentiels impactant tous les tenants.....	11
3.1.2.2.Purger les référentiels sur un seul tenant.....	11
3.1.3.Purger les journaux.....	12
3.1.4.Purger les Unités Archivistiques et les Groupes d’Objets.....	12
3.1.5.Purger les contrats.....	13
3.2.Rechercher et modifier un fichier.....	13
3.3.Ajouter et supprimer un parent.....	15
3.4.Test Audit correctif.....	16
3.5.Configuration de la plate-forme.....	16
3.6.Nettoyage d’un ingest corrompu.....	17
4.Tests.....	19
4.1.Tests de performance.....	19
4.1.1.Introduction.....	19
4.1.2.Champs disponibles.....	19
4.1.3.Résultats.....	19
4.2.Tests fonctionnels.....	21
4.2.1.Introduction.....	21
4.2.2.Lancer des tests Fonctionnels.....	21
4.2.3.Détail des tests.....	22
4.2.3.1.Partie « Résumé ».....	22
4.2.3.2.Partie « Détails ».....	23
4.3.Tests requêtes DSL.....	23
4.3.1.Introduction.....	23
4.3.2.Champs disponibles.....	24
4.3.3.Réaliser une requête.....	25
4.4.Visualisation du graphe.....	28
4.5.Test feature.....	29

5.Sécurisation des journaux.....	31
5.1.Lancer une opération de sécurisation.....	31
5.2.Journalisation des opérations de sécurisation.....	32
6.Tests manuels.....	33
6.1.Cahier de tests manuels.....	33
6.2.Requêtes DSL.....	33
7.Tests automatisés.....	34
7.1.Principes généraux.....	34
7.1.1.Tests de non régression.....	34
7.1.2.Tests fonctionnels.....	34
7.1.3.Behavior-Driven Development (BDD).....	35
7.2.Pré-Requis.....	35
7.2.1.Dépôt vitam-itest.....	35
7.2.2.Git LFS.....	35
7.3.Méthodologie de test.....	35
7.3.1.Séquencement.....	35
7.3.2.Lancement complet des TNR.....	36
8.Écriture des TNR.....	37
8.1.Structure des répertoires.....	37
8.2.Fichiers de Configuration.....	37
8.2.1.Nommage des fichiers.....	37
8.2.2.Informations transverses.....	37
8.3.Écriture d'un scénario.....	38
8.3.1.Structure d'un scénario.....	38
8.3.2.Insérer une requête DSL.....	39
8.3.3.Insérer un tableau.....	39
8.4.Annexes.....	40
8.4.1.Liste des actions d'étapes disponibles.....	40
8.4.2.Liste des fonctions disponibles.....	40
9.Guide d'écriture des tests Cucumber.....	42
9.1.Guide technique.....	42
9.1.1.Fonctionnalité : ingest.....	42
9.1.2.Fonctionnalité : recherche simple des métadonnées des unités archivistiques et des groupes d'objets techniques.....	43
9.1.3.Fonctionnalité : recherche complexe d'une unité archivistique.....	44
9.1.4.Fonctionnalité : recherche d'un registre des fonds.....	45
9.2.Scénarios fonctionnels.....	46
9.2.1.Collection Unit.....	46
9.2.2.Collection FileRules.....	48
9.2.3.Scénario Vérification et import des règles OK, recherche par «id» : OK.....	48
9.2.4.Collection AccessAccessionRegister.....	49
9.2.5.Tests de stockage.....	49
10.Tests curl.....	51
10.1.Introduction.....	51

10.2.API Externes.....	51
10.2.1.Ingest.....	51
10.2.2.Access.....	51
10.2.3.Administration fonctionnelle.....	52
10.3.Notes.....	52

## **1. OBJECTIF DU DOCUMENT**

Ce document présente d'une part les fonctionnalités développées sur l'IHM recette et d'autre part 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.

Plusieurs 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 automatisés permettent de vérifier de manière régulière qu'une régression n'est pas survenue et que tout fonctionne correctement.

## 2. PRINCIPES GÉNÉRAUX

### 2.1. Principes généraux

***Avertissement : L'IHM de recette est développée à des fins de test uniquement. Elle n'a aucunement vocation à être utilisée en production.***

L'IHM de recette contient des interfaces « utilisateur » permettant, par tenant, de :

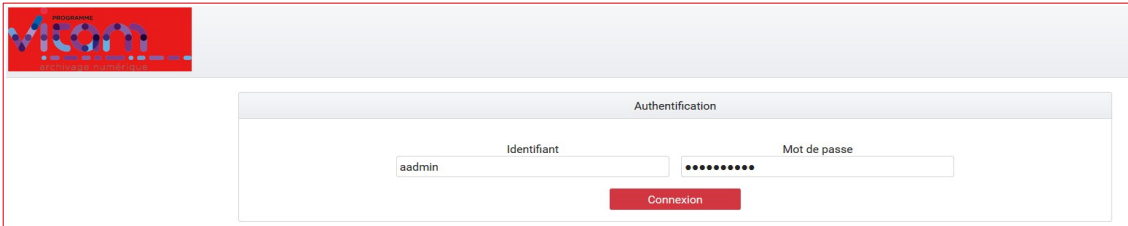
- Administrer les collections MongoDB (référentiels, journaux, objets, etc.),
- Lancer des tests (performance, fonctionnels, requêtes DSL, automatisés..., etc.),
- Sécuriser manuellement les journaux des opérations.

### 2.2. Accès

L'accès à l'IHM de recette s'effectue par un chemin différent de l'IHM de démonstration. Par défaut, son adresse est :

*adresse\_de\_votre\_serveur/ihm-recette/#/admin/collection*

Contrairement à l'IHM de démonstration, la sélection de tenant se fait après connexion. La page de connexion ne contient donc que les champs « Identifiant » et « Mot de passe ».



Par souci de distinction visuelle avec l'interface de démonstration, la couleur dominante de cette IHM est le rouge.

### 2.3. Navigation

Par défaut, suite à sa connexion l'utilisateur accède à la page d'administration des collections.

Le menu de navigation contient trois menus :



Les pages accessibles sont réparties de la façon suivante :

### Admin

- Administration des collections
- Recherche et Modification d'un fichier
- Ajout et suppression d'un parent
- Test audit correctif
- Configuration de la plate-forme
- Nettoyage d'un ingest corrompu

### Tests

- Tests de performance
- Tests fonctionnels
- Tests requêtes DSL
- Visualisation du graphe
- Test feature

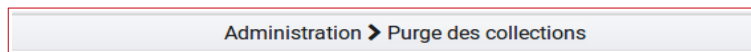
### Sécurisation

- Sécurisation des journaux

## 2.4. Fil d'Ariane

Le fil d'Ariane permet de visualiser le chemin d'accès à la page affichée. Il est situé en dessous du menu sur toutes les pages.

Il est composé du nom du menu dans lequel se trouve la page en cours, puis de la page consultée.

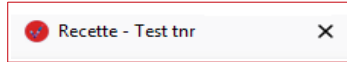


## 2.5. Titre des onglets

Sur l'IHM recette, le titre des pages est celui du dernier nœud du fil d'Ariane, précédé du mot « Recette -> ». Par exemple :

- Recette – Test Fonctionnels
- Recette – Administration des collections

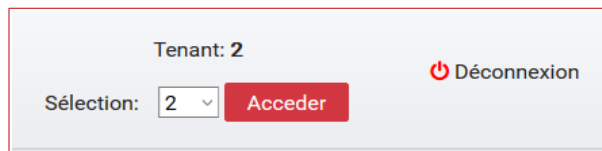




## 2.6. Sélection d'un tenant

Lors de la connexion, l'utilisateur n'est positionné sur aucun tenant. De ce fait, ses actions d'administration sont restreintes car celles-ci sont pour la plupart liées à un tenant. Par défaut, un certain nombre de boutons sont donc grisés et inactifs.

Pour sélectionner un tenant, il suffit de choisir celui désiré dans le menu déroulant en haut à droite de l'écran et de valider sa sélection en cliquant sur le bouton « Accéder ».



Une fois le tenant sélectionné, les boutons précédemment grisés sont activés et l'intégralité de l'interface de recette est disponible.

Dans le reste de ce document, il est considéré que l'utilisateur s'est placé dans le tenant sur lequel il veut effectuer ses opérations. L'utilisateur peut changer de tenant à tout moment, en répétant l'opération précédente.

**Note :** Les référentiels des formats, des contextes, des ontologies externes et des griffons sont liés à la plate-forme et non à un tenant. L'option de suppression de ces référentiels est toujours disponible, même si aucun tenant n'est sélectionné.

## 3. ADMINISTRATION

### 3.1. Administration des collections

L'administration des collections permet de supprimer certains ou tous les référentiels / journaux / objets / contrats dans un but de recette de la solution logicielle Vitam ou de faire des tests variés pour éprouver la stabilité du système.

L'utilisateur y accède par le menu, en cliquant sur « Administration des collections », ou par défaut lors de sa connexion.

Chaque collection comporte un bouton « Purger ». Lors du clic sur ce bouton « **Purger** », une fenêtre modale apparaît et demande de confirmer l'action de suppression.

Il existe deux types de purges.

#### 3.1.1. Purger toutes les collections de la solution logicielle Vitam

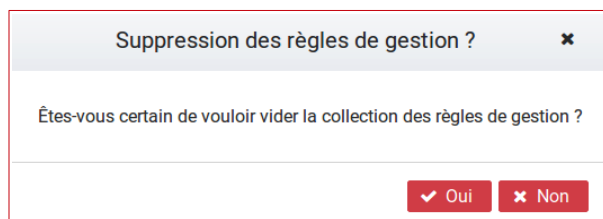
La purge de toutes les collections correspond à la suppression de tous les référentiels, contrats et journaux ainsi que de tous les objets et unités archivistiques à l'exception du référentiel des formats, des contextes, des ontologies internes et des griffons.

Suite à cette opération, chaque IHM correspondante est vide de contenu et plus aucune archive n'est présente dans la solution logicielle Vitam.

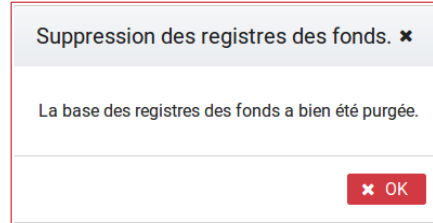
#### 3.1.2. Purger les référentiels

Il est possible de supprimer isolément un référentiel. Pour cela, il faut choisir un référentiel et cliquer sur « Purger » :

- Un clic sur la croix de la fenêtre modale ou sur « Non », annule la demande de suppression
- Un clic sur « Oui », valide la demande de suppression, la fenêtre modale se ferme et la suppression est effectuée



Une fois la suppression effectuée, un message de confirmation s'affiche dans une fenêtre modale.



### 3.1.2.1. Purger les référentiels impactant tous les tenants

- **Purger le référentiel des formats**

Le référentiel des formats de la solution logicielle Vitam est supprimé **pour tous les tenants**. L'IHM du référentiel de formats est vide de contenu. Sans référentiel des formats, aucun SIP ne pourra être importé dans la solution logicielle Vitam.

- **Purger les contextes applicatifs**

Lors de son exécution, la fonctionnalité de purge des contextes contrôle qu'il y a plus d'un contexte dans le référentiel. Si tel n'est pas le cas, la purge n'est pas réalisée. Si le référentiel contient plus d'un contexte, ils sont supprimés de la solution logicielle Vitam à l'exception de celui nommé « admin-context ».

- **Purger les ontologies externes**

La purge des ontologies externes supprime les ontologies présentes dans le système Vitam **pour tous les tenants**. Les vocabulaires internes (embarqués avec la solution) et externes sont supprimés. Les vocabulaires internes sont automatiquement réimportés. Cette fonctionnalité permet de revenir à l'état initial concernant le référentiel des ontologies avant l'ajout de vocabulaires externes.

- **Purger le référentiel des griffons**

Le référentiel des griffons de la solution logicielle Vitam est supprimé **pour tous les tenants**. L'IHM du référentiel des griffons est vide de contenu. Sans référentiel des griffons aucune action de préservation ne peut être réalisée.

### 3.1.2.2. Purger les référentiels sur un seul tenant

- **Le référentiel des règles de gestion**

Le référentiel des règles de gestion de la solution logicielle Vitam est supprimé pour le tenant sélectionné par l'utilisateur. L'IHM de démonstration du référentiel des règles de gestion est vide de contenu. Sans référentiel des règles de gestion, aucun SIP comportant des règles de gestion ne pourra être importé sur le tenant concerné dans la solution logicielle Vitam.

- **Le registre des fonds**

Le contenu du registre des fonds de la solution logicielle Vitam est supprimé pour le tenant sélectionné par l'utilisateur. L'IHM de démonstration du « registre des fonds » est vide de contenu.

- **Les profils d'archivage**

Tous les profils sont supprimés de la solution logicielle Vitam pour le tenant sélectionné par l'utilisateur. L'IHM de démonstration du « référentiel des profils » est vide de contenu.

- **Les services agents**

Le référentiel des services agents de la solution logicielle Vitam est supprimé pour le tenant sélectionné par l'utilisateur. L'IHM de démonstration du référentiel des services agents est vide de contenu. Sans référentiel de service agents, aucun SIP ne pourra être importé sur le tenant dans la solution logicielle Vitam.

- **Les profils d'unités archivistiques**

Le référentiel des profils d'unités archivistiques (AUP) de la solution logicielle Vitam est supprimé pour le tenant sélectionné par l'utilisateur. L'IHM de démonstration du référentiel des profils d'unités archivistiques (AUP) est vide de contenu.

- **Les scénarios de préservation**

Le référentiel des scénarios de préservation de la solution logicielle Vitam est supprimé pour le tenant sélectionné par l'utilisateur. L'IHM de démonstration du référentiel des scénarios de préservation est vide de contenu.

### **3.1.3. Purger les journaux**

- **Journal du cycle de vie (unités archivistiques)**

Tous les journaux du cycle de vie des unités archivistiques sont supprimés de la solution logicielle Vitam pour le tenant sélectionné par l'utilisateur. L'IHM de démonstration de chaque « Journal du cycle de vie » d'une unité archivistique est vide de contenu.

- **Journal du cycle de vie (groupes d'objets)**

Tous les journaux du cycle de vie des objets sont supprimés de la solution logicielle Vitam pour le tenant sélectionné par l'utilisateur. L'IHM de démonstration de chaque « Journal du cycle de vie » d'un objet est vide de contenu.

- **Journaux des opérations**

Tous les journaux des opérations sont supprimés de la solution logicielle Vitam pour le tenant sélectionné par l'utilisateur. Les IHM de démonstration « Journal des opérations » et « Journal des opérations d'entrées » sont vides de contenu.

### **3.1.4. Purger les Unités Archivistiques et les Groupes d'Objets**

- **Purger les Unités Archivistiques**

Toutes les unités archivistiques sont supprimées de la solution logicielle Vitam pour le tenant sélectionné par l'utilisateur. L'IHM de démonstration « Recherche d'archives » ne retourne plus d'unité archivistique.

- **Purger les groupes d'objets**

Tous les objets sont supprimés de la solution logicielle Vitam pour le tenant sélectionné par l'utilisateur. Les IHM « Recherche d'archives » et « Détail d'une unité archivistique » ne retournent plus de résultats.

*Note :* Lors de la purge de ces deux collections, l'utilisateur doit également penser à purger les journaux du cycle de vie des unités archivistiques et des groupes d'objets techniques.

### 3.1.5. Purger les contrats

- **Contrats d'accès**

Tous les contrats d'accès sont supprimés de la solution logicielle Vitam pour le tenant sélectionné par l'utilisateur. L'IHM de démonstration « Contrats d'accès » est vide de contenu.

- **Contrats d'entrée**

Tous les contrats d'entrée sont supprimés de la solution logicielle Vitam pour le tenant sélectionné par l'utilisateur. L'IHM de démonstration « Contrats d'entrée » est vide de contenu.

- **Contrats de gestion**

Tous les contrats de gestion sont supprimés de la solution logicielle Vitam pour le tenant sélectionné par l'utilisateur. L'IHM de démonstration « Contrats de gestion » est vide de contenu.

## 3.2. Rechercher et modifier un fichier

- **Rechercher un fichier**

Il est possible de rechercher un fichier grâce à son identifiant. L'identifiant à utiliser pour la recherche correspond à l'identifiant de l'objet que l'on peut récupérer via l'ihm de démonstration.

Par exemple pour rechercher l'objet binaire dont l'identifiant est :  
**aeaaaaaaaaahgynv2aar5ualoncyd4gaaaaba.**

Les éléments à saisir sont les suivants :

**Titre :** aeaaaaaaaaahgynv2aar5ualoncyd4gaaaaba

**Sur quelle Stratégie :** default

**Sur quelle Offre :** offer-fs-1.service.int.consul

**Catégorie :** Objet binaire

Puis cliquer sur « **Récupérer le fichier** ».

Un fichier portant le nom de l'identifiant est téléchargé.

Ce fichier peut alors être modifié avant d'être réimporté.

Administration > Recherche et modification d'un fichier

Recherche et Modification d'un fichier

Titre: aaaaaaaaaahh5o2aazsoalonu37ziaaaaaq

Sur quelle Stratégie: default

Sur quel Offre: offer-fs-1.service.int.consul

Catégorie: Objet binaires

Tenant: 0

Récupérer le fichier Supprimer le fichier

Exporter le fichier Vérifier état export

Modifier le fichier

Aucun emplacement cible choisi, merci de remplir le formulaire ci-dessus et récupérer le fichier original

sélectionner un fichier

ou

Glisser un fichier ici

Aucun fichier choisi

Importer

- **Modifier un fichier**

Une fois le fichier téléchargé, il est possible de le modifier, et de l'importer à nouveau. La version importée, remplacera la dernière version exportée.

Pour importer le fichier modifié :

Faire un glisser / déposer du fichier ou le sélectionner puis cliquer sur le bouton « **Importer** ».

Administration > Recherche et modification d'un fichier

Recherche et Modification d'un fichier

Titre: aeaaaaaaaaahnh5o2aazsoalonu37ziaaaaaq

Sur quelle Stratégie: default

Sur quel Offre: offer-fs-1.service.int.consul

Catégorie: Objet binaires

Tenant: 0

Récupérer le fichier Supprimer le fichier

Exporter le fichier Vérifier état export

Modifier le fichier

Modification du fichier: OBJECT - aeaaaaaaaaahnh5o2aazsoalonu37ziaaaaaq (Tenant: 0)

sélectionner un fichier

ou

Glisser un fichier ici

Nom du fichier: aeaaaaaaaaahnh5o2aazsoalonu37ziaaaaaq

Importer

**Note :** Les boutons « Exporter le fichier » et « Vérifier état export » ne peuvent être utilisés que lorsque la solution est implémentée avec une offre froide.

### 3.3. Ajouter et supprimer un parent

Il est possible d'ajouter ou de supprimer un lien entre deux unités archivistiques présentes dans la solution.

Pour cela, il faut rentrer les identifiants des unités archivistiques dans les champs correspondants, sélectionner un type d'opération « **Ajouter un lien** » ou « **Supprimer un lien** », sélectionner le contrat d'accès permettant d'avoir les droits pour les différentes modifications et cliquer sur le bouton « **Effectuer l'opération** ».

The screenshot shows a web form titled 'Ajout et suppression d'un parent' under the 'Administration' menu. The form contains the following fields and controls:

- Identifiant de l'unité archivistique parent: A text input field.
- Identifiant de l'unité archivistique enfant: A text input field.
- Opération: A dropdown menu with the text 'Sélectionner une action'.
- Tenant: A label 'Veillez choisir un tenant'.
- Contrat: A dropdown menu with the text 'Sélectionner un contrat'.
- Effectuer l'opération: A red button at the bottom.

*Note :* Il est possible d'avoir une visualisation de modifications de liens dans la section « visualisation du graphe ».

### 3.4. Test Audit correctif

Il est possible de lancer une opération de correction suite à un audit de cohérence qui aurait décelé une ou plusieurs erreurs de cohérence de données.

L'opération a pour but de remplacer les valeurs incorrectes détectées dans les offres de stockage ou dans la base de données. Pour cela, il faut sélectionner à la fois le tenant et l'identifiant de l'opération liée à l'audit de cohérence, puis cliquer sur le bouton « **Lancer la correction** ».

The screenshot shows a web form titled 'Correction d'audit' under the 'Administration' menu. The form contains the following fields and controls:

- Operation ID: A text input field with the placeholder 'Identifiant d'opération'.
- Tenant: A label 'Veillez choisir un tenant'.
- Lancer la correction: A red button at the bottom.

### 3.5. Configuration de la plate-forme

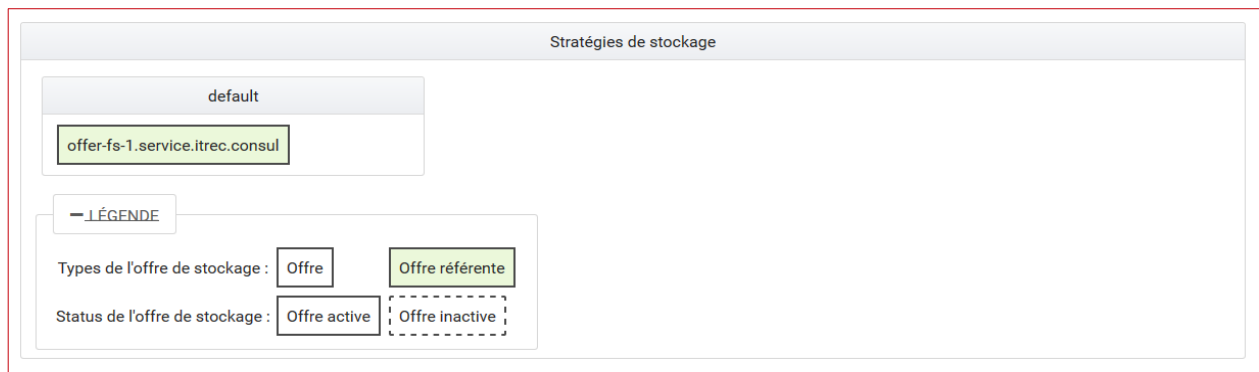
Il est possible de visualiser la configuration de la plate-forme et, en particulier, les stratégies de stockage paramétrées sur celle-ci.



Sont représentées les différentes stratégies de stockage, avec leur intitulé et les offres de stockage qu'elles référencent.

En cliquant sur le bouton « **Légende** », on accède à des explications relatives à la représentation de la stratégie de stockage et des offres qu'elle déclare :

- le type de l'offre de stockage :
  - l'offre est représentée par un rectangle vert si elle est référente,
  - l'offre est représentée par un rectangle transparent / blanc si elle n'est pas référente ;
- le statut de l'offre de stockage :
  - l'offre est active si elle est représentée par un rectangle sans pointillés,
  - l'offre est inactive si elle est représentée par un rectangle avec pointillés.



### 3.6. Nettoyage d'un ingest corrompu

Il est possible de lancer une opération de nettoyage d'entrée dont le processus s'est interrompu et ne peut être relancé. Cette opération a pu entraîner un enregistrement partiel des archives dans la solution logicielle Vitam.

L'opération a pour but de supprimer les archives et l'opération enregistrée dans le registre des fonds, si elle existe. Pour cela, il faut :

- sélectionner le tenant, l'identifiant de l'opération d'entrée en erreur et le contrat d'accès ;
- vérifier que l'opération d'entrée en erreur sélectionnée respecte les conditions à respecter dans le cadre de l'opération de nettoyage d'entrées corrompues et cliquer sur le bouton « radio » ;
- lancer l'opération au moyen du bouton « **Lancer le nettoyage** ».

**Lancement d'un ingest cleanup**

Operation d'ingest corrompu:

Tenant: 1

Contrat:  ▼

**– CONDITIONS D'ELIGIBILITÉ DES INGESTS À NETTOYER**

Cette procédure permet de nettoyer les données suite à un ingest incomplet / corrompu. Elle permet de purger toutes les unités archivistiques, groupes d'objets et objets binaires liés à l'ingest.

L'ingest à nettoyer doit satisfaire les conditions d'éligibilité suivantes :

- L'ingest n'est plus en cours d'exécution (RUNNING ou PAUSE).
- L'ingest s'est terminé avec une erreur (KO ou FATAL).
- Aucune unité d'un autre ingest n'a été rattachée en dessous d'une des unités de l'ingest à nettoyer.
- L'ingest à nettoyer n'a pas rajouté d'objets binaires à un group d'objets existant.
- Aucun autre ingest n'a rajouté d'objets binaires à l'un des groupes d'objets de l'ingest à nettoyer.
- L'ingest à nettoyer n'a pas rattaché une unité à un group d'objets existant.
- Aucun autre ingest n'a rattaché une autre unité à un groupe d'objets de l'ingest à nettoyer.

L'ingest respecte les conditions citées ?

## 4. TESTS

### 4.1. Tests de performance

#### 4.1.1. Introduction

Les tests de performance consistent à réaliser plusieurs fois l'entrée d'un SIP et à mesurer son temps d'exécution. Ces entrées peuvent être réalisées par une ou plusieurs tâches parallèles.

L'interface est accessible via le menu : Tests > Test de performance.

Les tests ne sont pas segmentés par tenant. Ces derniers sont directement configurés dans les tests. Il n'est donc pas nécessaire de sélectionner un tenant pour accéder au contenu de cette section.

#### 4.1.2. Champs disponibles

L'IHM est constituée de trois champs :

- Liste des SIP : liste des SIP disponibles pour réaliser le test. Ces SIP sont ceux déposés dans le dépôt vitam-itest. Il n'est possible de sélectionner qu'un SIP à la fois.
- Nombre de Thread : permet de définir le nombre de tâches parallèles qui exécuteront les entrées.
- Nombre d'Ingest : permet de définir le nombre total d'entrées à réaliser.





Un bouton « **Lancer les tests** » permet d'exécuter le test de performance.



#### 4.1.3. Résultats

Les résultats sont disponibles dans la section en bas de la page.

Chaque ligne représente un test de performance. Le nom du test est formaté de la façon suivante : report\_AAAAMMJJ\_HHmmSS.csv. Le bouton de téléchargement permet de récupérer le fichier .csv contenant les données du test.

Résultats	
Nom	Télécharger
report_20171013_160707.csv	
report_20171013_161326.csv	
report_20171014_042833.csv	
report_20171014_043626.csv	

Chaque ligne du fichier «.csv» représente une entrée. Les colonnes sont :

- OperationID
- PROCESS\_SIP\_UNITARY
- STP\_SANITY\_CHECK\_SIP
- SANITY\_CHECK\_SIP
- CHECK\_CONTAINER
- MANIFEST\_FILE\_NAME\_CHECK
- STP\_UPLOAD\_SIP
- STP\_INGEST\_CONTROL\_SIP
- CHECK\_SEDA
- CHECK\_HEADER
- CHECK\_HEADER.CHECK\_AGENT
- CHECK\_HEADER.CHECK\_CONTRACT\_INGEST
- PREPARE\_STORAGE\_INFO
- CHECK\_DATAOBJECTPACKAGE
- CHECK\_DATAOBJECTPACKAGE.CHECK\_MANIFEST\_DATAOBJECT\_VERSION
- CHECK\_DATAOBJECTPACKAGE.CHECK\_MANIFEST\_OBJECTNUMBER
- CHECK\_DATAOBJECTPACKAGE.CHECK\_MANIFEST
- CHECK\_DATAOBJECTPACKAGE.CHECK\_CONSISTENCY
- STP\_OG\_CHECK\_AND\_TRANSFORME
- CHECK\_DIGEST
- OG\_OBJECTS\_FORMAT\_CHECK
- STP\_UNIT\_CHECK\_AND\_PROCESS
- CHECK\_UNIT\_SCHEMA
- CHECK\_ARCHIVE\_UNIT\_PROFILE
- CHECK\_CLASSIFICATION\_LEVEL
- UNITS\_RULES\_COMPUTE
- STP\_STORAGE\_AVAILABILITY\_CHECK
- STP\_STORAGE\_AVAILABILITY\_CHECK.STORAGE\_AVAILABILITY\_CHECK

- STORAGE\_AVAILABILITY\_CHECK
- STORAGE\_AVAILABILITY\_CHECK.STORAGE\_AVAILABILITY\_CHECK
- STP\_OBJ\_STORING
- OBJ\_STORAGE
- OG\_METADATA\_INDEXATION
- STP\_UNIT\_METADATA
- UNIT\_METADATA\_INDEXATION
- STP\_OG\_STORING
- COMMIT\_LIFE\_CYCLE\_OBJECT\_GROUP
- OG\_METADATA\_STORAGE
- STP\_UNIT\_STORING
- COMMIT\_LIFE\_CYCLE\_UNIT
- UNIT\_METADATA\_STORAGE
- STP\_UPDATE\_OBJECT\_GROUP
- OBJECT\_LIST\_EMPTY
- STP\_ACCESSION\_REGISTRATION
- ACCESSION\_REGISTRATION
- STP\_INGEST\_FINALISATION
- ATR\_NOTIFICATION
- ROLL\_BACK

La colonne « Opération ID » contient le GUID de l'opération d'entrée. Les autres colonnes indiquent le temps en millisecondes qui a été nécessaire pour passer l'étape.

## 4.2. Tests fonctionnels

### 4.2.1. Introduction

La partie « Tests Fonctionnels » contient les écrans de lancement et de consultation des résultats des tests de non régression (ou TNR).

**NB** : La configuration des TNR ne s'effectue pas depuis ces écrans. La procédure de configuration est décrite dans le Chapitre 6 : « Tests Automatisés » du présent document.

Elle est accessible depuis le menu Tests > Test Fonctionnels.

Les tests ne sont pas segmentés par tenant. Ces derniers sont directement configurés dans les tests. Il n'est donc pas nécessaire de sélectionner un tenant pour accéder au contenu de cette section.

### 4.2.2. Lancer des tests fonctionnels

La page est divisée en deux parties :

- Tests fonctionnels
- Résultats des derniers tests



Le bouton « **Lancer les tests** » : permet de rejouer les tests configurés. Ceci donnera lieu à la création d'un nouveau rapport.

Le bouton « **Mise à jour référentiel** » : permet de récupérer les derniers fichiers de configuration des tests depuis « Git » (gestionnaire de sources). Ainsi, si un utilisateur a ajouté des tests et que ceux-ci ont été intégrés à Git, le fait de cliquer sur ce bouton permet de les prendre en compte au prochain clic sur le bouton « Lancer les Tests ».

Les résultats de derniers tests sont affichés dans un tableau à deux colonnes :

- Rapport
- Détail

Chaque ligne représente le rapport issu d'une campagne de tests. La colonne « Rapport » indique le nom du rapport. Celui-ci est constitué de la façon suivante : report\_AAAAMMJJ\_HHmms.json. Ainsi le rapport correspondant à la dernière campagne de tests se trouve en bas de la liste.

La colonne détail affiche simplement la mention « Accès au détail ».



Au clic sur une ligne, la page du détail du rapport concerné s'affiche sur l'écran.

### 4.2.3. Détail des tests

L'écran de détail d'une campagne de tests est divisé en deux parties :

- Partie Résumé
- Partie Détails

#### 4.2.3.1. Partie « Résumé »

La partie « Résumé » comporte les trois indications suivantes :

- Nombre de Tests : nombre de tests inclus dans la campagne
- Succès : nombre de tests en succès
- Échecs : nombre de tests en échec

RÉSULTATS DES DERNIERS TESTS		
Résumé		
Nombre de Tests	Succès	Echecs
164	153	11

#### 4.2.3.2. Partie « Détails »

Chaque ligne du tableau représente le résultat d'un test. La ligne est sur fond vert lorsque le test est en succès, sur fond rouge lorsqu'il est en échec.

Le tableau est constitué de quatre colonnes :

- Fonctionnalité : correspond à la fonctionnalité testée. Par défaut, un fichier de configuration correspond à une fonctionnalité. On a par exemple un fichier de configuration pour réaliser tous les tests sur l'INGEST. Dans ce cas, le nom de la fonctionnalité sera indiqué dans tous les cas de test correspondant dans le tableau de restitution.
- Identifiant : identifiant de l'opération correspondant au test. Il peut être utilisé pour trouver plus de détails sur le test dans le journal des opérations.
- Description : il s'agit d'une description du cas de test effectué. Elle est indiquée dans le fichier de configuration pour chacun des tests.
- Erreurs : erreur technique liée à l'échec du test. Cette colonne est vide pour les tests en succès.

Détails			
Fonctionnalité	Identifiant de l'opération	Description	Erreurs
uploader des fichier arbre et plan	aedqaaaaacffoklkabzuoak6fyadmxiar	Rattachement d'une unit d'arbre à une unit de sip	<pre> java.nio.file.AccessDeniedException: aeaaaaaaaaahg6wjfaa3beak6fyaeenvyaa at sun.nio.fs.UnixException.translateToIO at sun.nio.fs.UnixException.rethrowAsIOE at sun.nio.fs.UnixException.rethrowAsIOE at sun.nio.fs.UnixFileSystemProvider.new at java.nio.file.spi.FileSystemProvider.new at java.nio.file.Files.newOutputStream(Files.java:3016) at java.nio.file.Files.copy(Files.java:3016) </pre>

## 4.3. Tests requêtes DSL

### 4.3.1. Introduction

Le testeur de requêtes DSL met à disposition des administrateurs une interface graphique permettant de simplifier l'exécution de requêtes sur les API de la solution logicielle Vitam.

Celle-ci contient un formulaire composé de plusieurs champs.

### 4.3.2. Champs disponibles

**Tenant** : champ obligatoire. Indique le tenant sur lequel la requête va être exécutée. Ce champ est renseigné automatiquement avec le numéro du tenant sélectionné par l'administrateur.

**Contrat** : champ obligatoire. Liste permettant de sélectionner le contrat d'accès qui sera associé à la requête.

**Collection** : champ obligatoire. Liste permettant de sélectionner la collection sur laquelle la requête va être exécutée.

**Action** : champ obligatoire. Liste permettant de sélectionner le type d'action à effectuer.

- L'action « Rechercher » est possible pour l'ensemble des collections.
- L'action « Mettre à jour » est possible pour les collections suivantes :
  - Unités archivistiques
  - Profils d'archivage
  - Contrats d'accès
  - Contrats d'entrée
  - Contrats de gestion
  - Contextes applicatifs
- Les actions suivantes sont possibles pour la collection Opération :
  - Action - Suivant
  - Action - Pause
  - Action - Reprendre
  - Action - Stop

**Identifiant** : champ optionnel. Permet de renseigner le GUID de l'objet ciblé dans la collection.

**Requête DSL** : champ obligatoire. Permet de saisir la requête DSL au format JSON.



The screenshot shows the 'Requêtes DSL' interface. At the top, there is a breadcrumb 'Tests > Requêtes DSL'. Below it, there are two input fields: 'Tenant' with the value '8' and 'Identifiant' with the value 'Facultatif'. Underneath, there are three dropdown menus: 'Contrat' with the selected option 'Sélectionner un contrat', 'Collection' with 'Sélectionner une collection', and 'Action' with 'Sélectionner une action'. At the bottom, there are two large empty text areas labeled 'Requête DSL (format JSON)' and 'Réponse'.

### 4.3.3. Réaliser une requête

Pour réaliser une requête, l'administrateur remplit les champs du formulaire afin que leur contenu soit cohérent avec la requête qu'il souhaite exécuter.

The screenshot shows the 'Requêtes DSL' interface with the following values: 'Tenant' is '4', 'Identifiant' is 'Facultatif', 'Contrat' is 'ContratParDefaut', 'Collection' is 'Unit', and 'Action' is 'Rechercher'. The 'Requête DSL (format JSON)' field contains the following JSON query:

```
{
  "roots": [],
  "query": {
    "sort": [
      {
        "match": {
          "title": "Alma-Marceau"
        }
      },
      {
        "match": {
          "description": "Documentation"
        }
      }
    ],
    "depth": 20
  },
  "filter": {
    "orderby": {
      "transactedDate": 1
    }
  },
  "projection": {
    "fields": {
      "transactedDate": 1,
      "id": 1,
      "unittype": 1,
      "title": 1,
      "object": 1
    }
  }
}
```

At the bottom, there are three buttons: 'Valider JSON', 'Envoyer requête', and 'Effacer'.

Pour vérifier la validité du formatage du JSON, l'administrateur clique sur bouton « **Valider JSON** ». Si le « JSON » est valide, le texte est mis en forme et la mention « JSON Valide » est affichée à gauche du bouton. Dans le cas contraire, la mention « JSON non valide » est indiquée.

The screenshot shows a web interface for sending a request. On the left, there is a text area containing a JSON object: 

```
{ "TransactedDate": 1, "id": 1, "unittype": 1, "Title": 1, "#object": 1 }
```

. Below this text area are three buttons: "Valider JSON", "Envoyer requête", and "JSON non valide". On the right side of the interface, there is a button labeled "Effacer".

Pour exécuter la requête, l'administrateur clique sur le bouton « **Envoyer la requête** ». Le résultat est alors affiché à droite de l'écran dans la zone réponse. Il contient le retour envoyé par la solution logicielle Vitam.

The screenshot shows the search results interface. At the top, there are input fields for "Tenant" (value: 0) and "Identifiant" (value: Facultatif). Below these are three dropdown menus: "Contrat" (value: ContratTNR), "Collection" (value: Unit), and "Action" (value: Rechercher). The main area contains two side-by-side text areas displaying JSON responses. The left pane shows a search filter configuration: 

```
{ "$roots": [], "$query": [ { "$or": [ { "$match": { "Title": "Alma-Marceau" } }, { "$match": { "Description": "Documentation" } } ], "$depth": 20 }, "$filter": { "$orderby": { "TransactedDate": 1 } }, "$projection": { "$fields": {
```

. The right pane shows the search results: 

```
{ "statusCode": 200, "$hits": { "total": 137, "offset": 0, "limit": 10000, "size": 137 }, "$results": [ { "Title": "CR N° 32 GT DEM.doc", "id": "aeaqa...aagc6wdkab52oak7jbyoshiaaca", "#tenant": 0, "#object": "aebaaaaaagc6wdkab52oak7jbyon7qaaaaq", "#unitType": "INGEST", "#version": 0 }, { "Title": "ATT00003.png", "id": "aeaqa...aagc6wdkab52oak7jbyorpiaaafa", "#tenant": 0, "#object": "aebaaaaaagc6wdkab52oak7jbyomiaaaaba", "#unitType": "INGEST", "#version": 0 }, {
```

. Below the panes are buttons for "Valider JSON", "Envoyer requête", "JSON valide", and "Effacer".

Si la requête contient une erreur autre que le non-respect du formatage de la requête, le retour envoyé par la solution logicielle Vitam contiendra un code d'erreur et sera affiché de la façon suivante :

Tenant 0 Identifiant Facultatif

Contrat ContratTNR Collection Journal des opérations Action Rechercher

```
{
  "roots": [],
  "query": {
    "sort": [
      {
        "match": {
          "title": "Alma-Marceau"
        }
      },
      {
        "match": {
          "description": "Documentation"
        }
      }
    ],
    "depth": 20
  },
  "filter": {
    "orderby": {
      "transactedDate": 1
    }
  },
  "projection": {
    "fields": {
```

```
{
  "statusCode": 500,
  "code": "020104",
  "context": "External Access",
  "state": "Input / Output",
  "message": "Access external client error in selectOperation method.",
  "description": "Request precondition failed"
}
```

Valider JSON Envoyer requête JSON valide Effacer

Si la requête envoyée par l'administrateur ne respecte pas le formatage de la requête, l'endroit où se trouve l'erreur sera indiqué dans le retour de la façon suivante :

Tenant 0 Identifiant Facultatif

Contrat ContratTNR Collection Unit Action Rechercher

```
"match": {
  "title": "Alma-Marceau"
},
{
  "match": {
    "description": "Documentation"
  }
},
],
"depth": 20
},
"filter": {
  "orderby": {
    "transactedDate": 1,
  }
},
"projection": {
  "fields": {
    "transactedDate": 1,
    "#id": 1,
    "#unittype": 1,
    "title": 1,
    "#object": 1
  }
}
```

```
"Unexpected character (')' (code 125): was expecting double-quote to start field name\n at [Source: org.glassfish.jersey.message.internal.ReaderInterceptorExecutor$UnCloseableInputStream@2a19985a; line: 23, column: 6]"
```

Valider JSON Envoyer requête JSON non valide Effacer

L'utilisateur peut vider le contenu de l'espace dédié à la réponse du DSL en cliquant sur le bouton « **Effacer** ».

## 4.4. Visualisation du graphe

**Note :** L'écran utilisé est expérimental.

Cette partie permet d'avoir une représentation visuelle d'un graphe contenu dans un SIP. La première étape consiste donc à récupérer les informations suivantes :

- L'identifiant de l'opération (appuyer sur la touche « Entrer » du clavier pour créer un tag)
- L'intitulé du contrat utilisé

The screenshot shows a web interface titled 'Tests > Visualisation du Graphe'. At the top, there is a red banner with '!!! EXPERIMENTAL !!!'. Below it, there are two input fields: 'Contrat' with a dropdown menu and 'Identifiants des opérations' with a text input. To the right of the second field is a red button labeled 'Envoyer la requête'. The 'Contrat' dropdown is currently set to 'Sélectionner un contrat'.

Il faut ensuite rajouter les informations dans les champs prévus à cet effet : « Contrat » et « Identifiant d'opération ».

Puis il suffit de cliquer sur le bouton « **Envoyer la requête** » pour visualiser plusieurs choses :

- Sur la partie gauche, la représentation visuelle du graphe contenu dans le SIP
- Sur la partie droite, lorsqu'on clique sur la représentation de chaque unité archivistique, le détail des données liées à l'unité archivistique s'affiche

The screenshot shows the same interface as above, but now with data entered. The 'Contrat' dropdown is set to 'Contrat d'accès Musée du Quai Brai'. The 'Identifiants des opérations' field contains 'aeaaaaabchfj4pcaa2xialiquvgo3yaaaaq' with a red 'x' icon to clear it. The 'Envoyer la requête' button is now active. On the left, a graph visualization shows a hierarchy of nodes representing archival units. On the right, a panel titled 'Détail de l'Archive Unit' displays the following JSON metadata:

```

{
  "DescriptionLevel": "RecordGrp",
  "Title": "4_ Porte de Clignancourt",
  "Description": "Cette unité de description doit hériter de ManagementMetadata la règle ACC-00002 avec comme StartDate 01/01/2000 et a une règle propre DIS-00001 avec comme StartDate 01/01/2000",
  "StartDate": "2017-04-05T08:11:56",
  "EndDate": "2017-04-05T08:11:56",
  "#id": "aeqaaaaabahaffe6aafawaliquiodaaaaadq",
  "#tenant": 8,
  "#unitups": [],
  "#min": 1,
  "#max": 1,
  "#allunitups": [],
  "#management": {
    "AccessRule": {
      "Rules": [
        {
          "Rule": "ACC-00002",
          "StartDate": "2000-01-01",
          "EndDate": "2025-01-01"
        }
      ]
    }
  },
  "DisseminationRule": {
  }
}
    
```

## 4.5. Test feature

**Note :** Cette fonctionnalité est utilisée uniquement par l'équipe Vitam pour tester les TNR ( Test de Non Régression).

Cette interface permet d'exécuter des TNR sans devoir disposer d'un environnement local complet. Ces TNR seront exécutés sur l'environnement où se situe l'IHM recette.

Cette fonctionnalité nécessite l'existence de la branche Git nommée « tnr\_master » sur laquelle l'IHM recette va se brancher pour l'exécution.

La page est composé de deux champs de texte :

- Celui du haut prend en entrée des TNR à exécuter
- Celui du bas est la sortie et affiche la réponse à l'exécution

Cette page contient aussi 3 boutons :

- « **Mise à jour référentiel** » : récupère les commits de la branche « tnr\_master ». La mise à jour est utile lorsqu'un nouveau jeu de données est envoyé sur tnr\_master et que l'IHM-recette doit le récupérer
- « **Lancer le TNR** » :
  - Si l'IHM recette utilise la branche « master », alors elle change pour aller sur la branche tnr\_master et récupère son contenu. Puis, elle exécute le TNR écrit dans le premier champ de texte.
  - Si l'IHM recette est déjà branchée sur « tnr\_master », alors elle exécute directement le TNR écrit dans le premier champ de texte

Dans les deux cas le contenu de la réponse est effacé avant le lancement du TNR

- « **Effacer** » : supprime le contenu de la réponse

Programme Vitam – IHM Recette – v. 9.0

Tests ▶ Test tnr

Exécution manuelle de TNR

Mise à jour référentiel

Lancer le TNR

Effacer

## 5. SÉCURISATION DES JOURNAUX

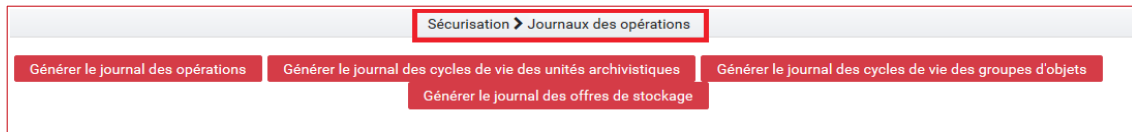
La sécurisation des journaux est une action visant à assurer la valeur probante de l'information prise en charge dans la solution logicielle Vitam.

### 5.1. Lancer une opération de sécurisation

Les opérations que l'on peut réaliser, sont représenté par les boutons suivants :

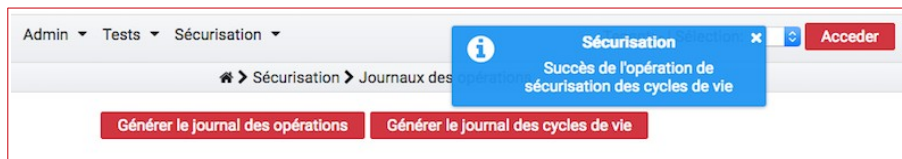
- « Générer le journal des opérations »
- « Générer le journal des cycles de vie des unités archivistiques »
- « Générer le journal des cycles de vie des groupes d'objets »
- « Générer le journal des offres de stockage »

Au clic sur un des boutons le système va lancer l'opération de sécurisation des journaux. Elle prendra en compte tous les journaux, du dernier créé au dernier non sécurisé. Un message s'affiche alors sur l'écran précisant le succès de l'opération.



Si aucun journal n'a encore été sécurisé, alors l'opération de sécurisation prendra en compte tous les journaux d'opération existant dans la solution logicielle Vitam.

À la fin de l'opération, un message avertit du succès ou de l'échec de l'opération.



Un fichier «.zip» est créé et placé dans l'offre de stockage de Vitam dans le répertoire suivant :

```
/browse/data/storage-offer-default/0/Logbook
```

Il contient les fichiers suivants :

- *operation.json* : liste des opérations sécurisées, la première étant l'opération « traceability »
- *merkleTree.json* : contient une sérialisation JSON de l'arbre de merkle

- *token.tsp* : horodatage de la combinaison de la racine de l'arbre de merkle, des empreintes des opérations de sécurisation antérieures (la dernière réalisée, celle du mois précédent et celle de l'année précédente)
- *computing\_information.txt* : reprend les différentes empreintes qui ont permis de réaliser l'horodatage
- *additional\_information.txt* : contient le nombre d'informations sécurisées, ainsi que les dates du premier et du dernier élément

## 5.2. Journalisation des opérations de sécurisation

La sécurisation des journaux des opérations donne lieu à la création d'un journal des opérations de type TRACEABILITY, consultable depuis l'IHM de démonstration.

Ces journaux sont créés par tenant.



## 6. TESTS MANUELS

**Les tests manuels peuvent être effectués :**

- À l'aide du cahier de tests manuels.
- Au travers de requêtes DSL

### 6.1. Cahier de tests manuels

Le cahier de test manuel se présente sous forme de tableur. Il répertorie tous les cas de tests 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

### 6.2. Requêtes DSL

Il est possible de lancer des requêtes DSL 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 (recherche ou 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 au format JSON dans le champ texte de droite.

Le Chapitre 3 : « Tests / III – Tests requêtes DSL » détaille l'utilisation des requêtes DSL

## 7. TESTS AUTOMATISÉS

### 7.1. Principes généraux

#### 7.1.1. Tests de non régression

L'objectif des Tests de Non Régression (TNR) est 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 tests de non régression permet 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 plate-forme.

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.

#### 7.1.2. Tests fonctionnels

Cucumber est l'outil de tests fonctionnels dans Vitam, il est accessible via 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.



**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



### 7.1.3. 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>).

## 7.2. Pré-Requis

### 7.2.1. Dépôt 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.

### 7.2.2. 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
```

## 7.3. Méthodologie de test

### **7.3.1. Séquencement**

Les tests sont regroupés par lot intellectuellement cohérent dans des fichiers « .feature ». Chaque fichier 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.

### **7.3.2. Lancement complet des TNR**

Il est possible de lancer tous les TNR 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 ».

## 8. ÉCRITURE DES TNR

### 8.1. Structure des répertoires

Le répertoire du dépôt vitam-itest est structuré 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.

### 8.2. Fichiers de Configuration

#### 8.2.1. 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'entrée
- un fichier pour les tests 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
```

#### 8.2.2. 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 correspondant à la langue utilisée pour les descriptions. Par exemple :

```
language: fr.
```

**Annotation** : information optionnelle permettant 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 permettant 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 unité archivistique existante
```

**Contexte** : information optionnelle contenant des actions qui vont s'exécuter pour chacun des scénarios. À ce titre, elles s'écrivent comme les actions d'un scénario. Le contexte doit être indenté de 1 par rapport aux autres éléments. Par exemple :

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

## 8.3. Écriture d'un scénario

### 8.3.1. 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
  Étant 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
```

### 8.3.2. Insérer une requête 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}}}
  ""
```

### 8.3.3. 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 métadonnées sont
| Title           | Liste des armements |
| DescriptionLevel | Item                 |
| StartDate       | 1917-01-01          |
```

EndDate	1918-01-01
---------	------------

## 8.4. Annexes

### 8.4.1. Liste des actions d'étapes disponibles

Les types d'actions sont les suivants :

- une situation initiale (les acquis) : **Étant 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**)

<i>Action</i>
Étant donné
Quand
Alors
Mais
Et

### 8.4.2. Liste des fonctions disponibles

<i>Fonctionnalité</i>	<i>Doit être suivi par</i>
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



<b>Fonctionnalité</b>	<b>Doit être suivi par</b>
le[s]? statut[s]? (? :de l'événement des événements) (.*) (?:est sont) (.*)	un ou plusieurs 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 paramètre et une valeur de remplacement
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 métadonnées sont (.*)	un tableau
les métadonnées 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 métadonnées 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 métadonnées pour le détail du registre des fonds sont	un tableau

## 9. 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.

### 9.1. Guide technique

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

#### 9.1.1. Fonctionnalité : ingest

**Scénario** : Envoi d'une archive et vérification du journal de l'opération, de l'ATR, des journaux de cycle de vie d'une unité archivistique et d'un groupe d'objets techniques.

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

```
Scénario: Guide ingest
# Exécution d'un ingest
  Étant 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
  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
```

### 9.1.2. Fonctionnalité : recherche simple des métadonnées des unités archivistiques et des groupes d'objets techniques

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

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

```
Scénario : Guide recherche simple
# Exécution d'un ingest
  Étant 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 métadonnées 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 |
  Quand je recherche les groupes d'objets de l'unité archivistique dont
le titre est ID8
```

```
Alors les métadonnées sont
  | #qualifiers.1.versions.0.DataObjectVersion |
BinaryMaster_1 |
  | #qualifiers.1.versions.0.FileInfo.FileName |
Filename0 |
  | #qualifiers.1.versions.0.FormatIdentification.FormatId |
fmt/18 |
```

### 9.1.3. Fonctionnalité : recherche complexe d'une unité archivistique

**Scénario** : Recherche d'une unité archivistique particulière et de son groupe d'objet.

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

```
Scénario: Guide recherche avancée
# Exécution d'un ingest
  Étant 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 métadonnées 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,  
    "Title": 1  
  }  
}  
}
```

#### 9.1.4. 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 présume 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  
# Exécution d'un ingest  
Étant 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" } },  
  "$projection": {}  
}  
""  
Et je recherche les registres de fond  
# Vérification du registre de fonds  
Et le nombre de registres de fond est 1  
Et les métadonnées pour le registre de fond sont  
| OriginatingAgency          | FRAN_NP_009913          |  
| TotalObjects.ingested        | 4 |  
| TotalObjectGroups.ingested   | 4 |  
| TotalUnits.ingested          | 7 |  
# Rechercher du détail du registre de fonds pour l'ingest  
Quand j'utilise la requête suivante  
""  
{  
  "$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 métadonnées pour le détail du registre de fond sont  
| OriginatingAgency | FRAN_NP_009913 |  
| TotalObjects.ingested | 4 |  
| TotalObjectGroups.ingested | 4 |  
| TotalUnits.ingested | 7 |
```

## 9.2. Scénarios fonctionnels

### 9.2.1. 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).

```
Étant 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 métadonnées 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).

```
Étant 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 métadonnées 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": [ { "$and": [ {
"$seq": { "#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 ».

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

**Fonctionnalité** Affichage des métadonnées de l'objet physique.

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

```
Étant 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": [{"$and":[{"$seq":{"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 métadonnées 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 |
```

### 9.2.2. Collection FileRules

**Fonctionnalité** Recherche de règle de gestion.

### 9.2.3. 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 métadonnées sont
| RuleId | APP-00001
""""
{ "$query": { "$eq": { "RuleId":
"APP-00001" } }, "$projection":
{ "$fields": { "#id": 1,
"RuleId": 1, "Name": 1 } },
"$filter": {} }
""""
```

### 9.2.4. Collection AccessAccessionRegister

**Fonctionnalité** Recherche dans les registres des fonds.

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

**Scénario** Téléchargement d'un SIP et vérification du contenu dans le registre de fonds.

```
Étant 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": {}
}
""""
```

### 9.2.5. 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

## 10. TESTS CURL

### 10.1. 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 (par exemple : remplacer {env.programmevitam.fr}).

### 10.2. API Externes

#### 10.2.1. 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 :

Exemple de requête curl pour faire un ingest :

```
curl -v -X POST -k --key vitam-vitam_2.key --cert vitam-vitam_2.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"}'
```

#### 10.2.2. Access

Exemple de requête curl pour mettre à jour une AU

```
curl -v -X PUT -k --key vitam-vitam_2.key --cert vitam-vitam_2.pem
'https://{env.programmevitam.fr}/access-external/v1/units/aeaqaahmtu
sqabz5oalc4p2zu5aaaaaq' -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":
```



8Hk+dle9lmg1sMlzHVcTVauCuvrk8WCec9ja56+b9N4JbaCwYFmMRlMzdBQU4LXrbqxlakp  
a2ua0mSzCKe8WHI9m5uCHhUi3fMa7KJsN5nBHkw63nFwGQwyRNQYgZiyhmzXtez/  
l+8f1quMAPoTIIlsG+TBFW0s9+LqY8ufE9+8u8S1FynZlsgfIoKl2bKVXWwrZVfJ+S8mh6mH  
4V3MuhLwljv+/6HDZCc3FoY5eN/  
lyWI49Maz5W87bKqNyecYtrBlvML7k5UeOLtgNuUsTBlzFTxMkaQHOSpMyrHZ/  
yVPNVfuP3cCKvzMPHFGHzJZK0qvz4zdFdx7YzBq+I6YLvRES9b+DkvdrTOpZI2GjKuP5m13  
kcUjsFeqJR6rb+o1kJuCj/QMC2OjMXMlDqNa8mL5ooGQmYOzHkfq4vdKLG/  
Fvbpw2DDrww9jKmw2l6eWLYzuIpvz7sqUHwi30wScXSm/FCKF9DjzODUpSkBvDiaA== '