Module de collecte

Date	Version
27/01/2023	2.0 (Version 6.RC)

État du document

◯ En projet ◯ Vérifié ◉ Validé

Maîtrise du document

Responsabilité	Nom	Entité	Date
Rédaction	MVI	Équipe Vitam	29/03/2022
Vérification	MVI	Équipe Vitam	20/07/2022
Validation	AGR	Équipe Vitam	27/01/2023

Suivi des modifications

Version	Date	Auteur	Modifications
0.1	29/03/2022	MVI	Initialisation
0.2	16/05/2022	IJO	Relecture
0.3	19/05/2022	MVI	Corrections
0.4	31/05/2022	Equipe Vitam	Relecture
0.5	10/06/2022	MVI	Corrections
1.0	27/06/2022	AGR	Finalisation du document pour publication de la Version 5
1.1	20/01/2023	MVI	Mise à jour pour tenir compte des fonctionnalités mises en œuvre pendant la Version 6.RC : • Section 2.1 (« Qu'est-ce que la collecte d'archives ? ») : ajout de la notion de contrat et de projet de versement ; • Section 2.2 (« Qu'est-ce que le module de collecte ? ») : implémentation de l'APP « Collecte et préparation des versements » ; ajout de fonctionnalités de paramétrage d'un versement, de consultation des (pré-)versements, de modification des métadonnées, de gestion des statuts, de suppression de projets et de (pré-)versements ; • Section 2.3 (« Pourquoi, quand et comment utiliser le module de collecte ?») : implémentation de l'APP « Collecte et préparation des versements » ; possibilité de mettre à jour les métadonnées ; • Section 3.1 (« Configuration») : ajout de la section ; • Section 3.2 (« Versement») : changement du path de l'API « collect » en « collect-external » ; ajout d'un projet en prérequis de la création d'une transaction (sous-section 3.2.2) ; ajout de champs dans la transaction (sous-section 3.2.2.1) ; ajout

			des sous-sections 3.2.3 « Utilisation des API pour des envois en lot » et 3.2.4 « Utilisation dans VitamUI » ; • Section 3.3 (« Accès») : ajout de la section ; • Section 3.4 (« Gestion des archives») : ajout de la section ; • Section 3.5 (« Transfert») : changement du path de l'API « collect » en « collect-external » ; changement du statut « CLOSE » et ajout de nouveaux statuts et d'une purge automatique ; ajout de la sous-section 3.5.3 ; • Section 4.2 (« Comment sont gérés les statuts d'une transaction ») : ajout de la sous-section ; • Annexe 3 (« Liste des points d'API ») : ajout de l'annexe.
2.0	27/01/2023	AGR	Finalisation du document pour publication de la Version 6.RC

Documents de référence

Document	Date de la version	Remarques
NF Z 44022 – MEDONA – Modélisation des données pour l'archivage	18/01/2014	
Standard d'échange de données pour l'archivage – SEDA – v. 2.1	06/2018	
Standard d'échange de données pour l'archivage – SEDA – v. 2.2	02/2022	Cette nouvelle version du SEDA est intégrée à la solution logicielle Vitam à partir de la V6.RC.
Vitam – Structuration des *Submission Information Package* (SIP)	27/01/2023	
Vitam – Modèle de données	27/01/2023	
Vitam – Ontologie	27/01/2023	
Vitam – Profils d'archivage	27/01/2023	
Vitam – Profils d'unité archivistiques	27/01/2023	

Licence

Table des matières

1. Résumé

Jusqu'à présent, pour la gestion, la conservation, la préservation et la consultation des archives numériques, les acteurs du secteur public étatique ont utilisé des techniques d'archivage classiques, adaptées aux volumes limités dont la prise en charge leur était proposée. Cette situation évolue désormais rapidement et les acteurs du secteur public étatique doivent se mettre en capacité de traiter les volumes croissants d'archives numériques qui doivent être archivés, grâce à un saut technologique.

1.1. Présentation du programme Vitam

Les trois ministères (Europe et Affaires étrangères, Armées et Culture), combinant légalement mission d'archivage définitif et expertise archivistique associée, ont décidé d'unir leurs efforts, sous le pilotage de la Direction interministérielle du numérique (DINum), pour faire face à ces enjeux. Ils ont décidé de lancer un programme nommé Vitam (Valeurs Immatérielles Transmises aux Archives Pour Mémoire) qui couvre plus précisément les opérations suivantes :

- la conception, la réalisation et la maintenance mutualisées d'une solution logicielle d'archivage électronique de type back-office, permettant la prise en charge, le traitement, la conservation et l'accès aux volumes croissants d'archives (projet de solution logicielle Vitam) ;
- l'intégration par chacun des trois ministères porteurs du Programme de la solution logicielle dans sa plate-forme d'archivage. Ceci implique l'adaptation ou le remplacement des applications métiers existantes des services d'archives pour unifier la gestion et l'accès aux archives, la reprise des données archivées depuis le début des années 1980, la réalisation d'interfaces entre les applications productrices d'archives et la plate-forme d'archivage (projets SAPHIR au MEAE, ADAMANT au MC et ArchiPél au MinArm) ;
- le développement, par un maximum d'acteurs de la sphère publique, de politiques et de plates-formes d'archivage utilisant la solution logicielle.

La solution logicielle Vitam est développée en logiciel libre et recourt aux technologies innovantes du Big Data, seules à même de relever le défi de l'archivage du nombre d'objets numériques qui seront produits ces prochaines années par les administrations de l'État. Afin de s'assurer de la qualité du logiciel livré et de limiter les dérives calendaires de réalisation, le projet est mené selon une conduite de projet Agile. Cette méthode dite « itérative », « incrémentale » et « adaptative » opère par successions de cycles réguliers et fréquents de développements-tests-corrections-intégration. Elle associe les utilisateurs tout au long des développements en leur faisant tester les éléments logiciels produits et surtout en leur demandant un avis sur la qualité des résultats obtenus. Ces contrôles réguliers permettent d'éviter de mauvaises surprises lors de la livraison finale de la solution logicielle en corrigeant au fur et à mesure d'éventuels dysfonctionnements.

Le programme Vitam a bénéficié du soutien du Commissariat général à l'investissement dans le cadre de l'action : « Transition numérique de l'État et modernisation de l'action publique » du

Programme d'investissement d'avenir (PIA). Il a été lancé officiellement le 9 mars 2015, suite à la signature de deux conventions, la première entre les ministères porteurs et les services du Premier ministre, pilote du programme au travers de la DINum, et la seconde entre les services du Premier ministre et la Caisse des dépôts et consignations, relative à la gestion des crédits attribués au titre du Programme d'investissements d'avenir.

La phase projet du Programme Vitam s'est achevée début 2020 avec la publication de la V3 de la solution logicielle et le lancement de la phase produit, définie par une convention de maintenance et amélioration continue entre les ministères porteurs et les services du Premier ministre. Cette nouvelle phase maintient le pilotage stratégique interministériel et confie le pilotage opérationnel au ministère de la Culture. La place des utilisateurs est renforcée par la création du Club utilisateurs, dont un représentant participe aux instances de gouvernance et qui a vocation à permettre les échanges, les retours d'expériences, l'entraide, la définition d'évolution, les contributions, etc.

Chaque version de la solution logicielle Vitam est maintenant composée du back-office, enrichi par un front-office développé par des utilisateurs et nommé Vitam UI. Une version « release candidate » est dorénavant publiée à l'automne, avant la publication d'une nouvelle version majeure au printemps : fin 2022, le Programme Vitam met ainsi à disposition de tous sa version 6.RC.

1.2. Présentation du document

Le document présente les fonctionnalités associées à l'utilisation du module de collecte dans la solution logicielle Vitam (version *béta*).

Il s'articule autour des axes suivants :

- une présentation du module de collecte ;
- une présentation des mécanismes mis en œuvre dans la solution logicielle Vitam pour prendre en compte les opérations de collecte, en application du SEDA ;
- des conseils de mise en œuvre.

Le présent document décrit les fonctionnalités qui sont offertes par la solution logicielle Vitam au terme de la version 6.RC (fin 2022). Il a vocation à être amendé, complété et enrichi au fur et à mesure de la réalisation de la solution logicielle Vitam et des retours et commentaires formulés par les ministères porteurs et les partenaires du programme.

2. Présentation du module de collecte

2.1. Qu'est-ce que la collecte d'archives ?

La **collecte** d'archives désigne l'ensemble des opérations qui vont conduire au **transfert** d'un ensemble d'archives :

- d'un service producteur à un service d'archives,
- d'un service d'archives intermédiaires à un service d'archives définitives.

Un **versement** correspond à un ensemble d'archives faisant l'objet du transfert.
Le **transfert** matérialise l'opération visant à transférer la responsabilité de la conservation des archives d'un service à un autre.

La collecte d'archives peut intervenir à plusieurs occasions :

- Un service producteur soumet un versement à un service d'archives ;
- La collecte peut être à l'initiative d'un service d'archives, dans le cadre d'une campagne de collecte d'archives ;
- Des règles de gestion associées à des archives arrivent à échéance et impliquent un transfert de responsabilité en termes de conservation de ces dernières et, de fait, un transfert vers un autre service d'archivage.

La manière de procéder au transfert d'archives est définie dans la norme NF Z 44-022 et dans sa déclinaison pour les acteurs du service public, le Standard d'échanges de données pour l'archivage (SEDA).

Le versement, pouvant également être nommé transfert ou **transaction**, s'effectue en deux temps :

- la soumission d'un ensemble d'archives à verser en vue de :
  - vérifier son contenu et, le cas échéant, procéder à des traitements (ex. suppression de documents, renommage, etc.) ;
  - documenter et contextualiser le versement, par l'ajout d'informations sur le service producteur, son contenu, ses règles de gestion ;
- après validation de l'ensemble d'archives candidates à l'archivage par les parties concernées, transfert des archives en vue d'en assurer leur conservation et transfert de la responsabilité liée à leur conservation.

Au préalable de versements, une politique ou stratégie de versement est définie par le service d'archives, en termes de rôles et responsabilités, mais aussi de contenu attendu. Elle est définie dans un contrat de versement, dont une partie est matérialisée dans un projet de versement et un contrat d'entrée.

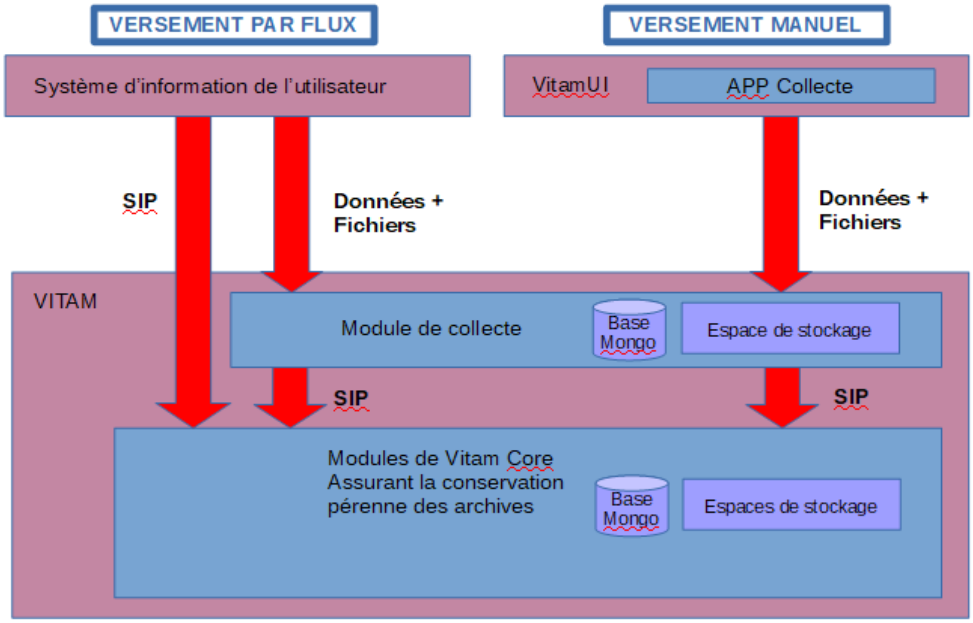
2.2. Qu'est-ce que le module de collecte ?

La solution logicielle Vitam met à disposition un **module de collecte** en vue de recevoir des ensembles hétérogènes d'archives, de paramétrer leur réception, de procéder à des traitements sur ces archives et de les transférer pour conservation dans le système d'archivage électronique. Ce module :

- constitue un service du back-office de la solution logicielle Vitam ;
- est utilisé par l'APP « Collecte et préparation des versements » du front-office VitamUI.

Ce module intervient en amont du transfert (ou versement) d'archives dans la solution logicielle Vitam. Son installation, ainsi que son utilisation, sont optionnelles.

Positionnement du module de collecte en amont du versement



Il permet de :

Configurer des versements	Définir un projet de versement et un rattachement automatisé à une position dans l'arborescence.
(Pré-)Verser les archives	Collecter depuis l'extérieur un ensemble d'archives, caractérisées par des métadonnées et des fichiers numériques, et constituer : • des (pré-)versements (ou transactions) automatisés émanant d'un système d'information externe ; • des (pré-)versements (ou transactions) manuels et unitaires ◦ constitués par exemple d'arborescences bureautiques ou de messageries,

	○ réalisés depuis des interfaces, notamment celles de l'APP « Collecte et préparation des versements » du front-office VitamUI
Consulter les (pré-)versements	Consulter : • la liste des projets de versement et des (pré-)versements (ou transactions) en attente, • un projet de versement en particulier, c'est-à-dire sa description, les informations contextuelles et la position de rattachement dans le tenant de destination, • le contenu d'un (pré-)versement (ou transaction), c'est-à-dire la liste des archives associées, une unité archivistique en particulier et, le cas échéant, l'objet numérique associé.
Traiter les archives	Procéder à des traitements archivistiques tels que : • définition de métadonnées contextuelles , • identification de format, • calcul d'empreintes, • calcul du poids de l'objet numérique, • réorganisation d'arborescence *(service non implémenté)*, • mise à jour de métadonnées descriptives et de gestion, • tri, dédoublonnage, suppression unitaire d'objets numériques *(services non implémentés)*, • gestion de statuts (ex. réouverture d'un (pré-)versement en erreur), • suppression de projets et de (pré-)versements, • etc.
Transférer les archives	• Générer un SIP conforme au Standard d'échanges de données pour l'archivage (SEDA) et le transférer dans le système d'archivage électronique pour conservation. • Suppression automatisée d'un (pré-)versement

2.3. Pourquoi, quand et comment utiliser le module de collecte ?

Le module de collecte permet de :

- faciliter, voire automatiser, les versements (ou transactions) d'archives, depuis un service externe vers la solution logicielle Vitam ;
- le cas échéant, effectuer un contrôle supplémentaire sur les projets de versement et leur contenu, en vue de vérifier la qualité des données, en amont de leur transfert dans un système d'archivage électronique ;
- enrichir les métadonnées (notamment par l'identification de formats, le calcul d'empreintes et du poids de l'objet numérique, ou encore la mise à jour des métadonnées descriptives et de gestion) ;

- proposer un format pivot et générique pour les services externes souhaitant verser des archives au sein de la solution logicielle Vitam ;
- améliorer l'interopérabilité entre les systèmes d'information.

L'utilisation du module de collecte peut être envisagée dans les cas suivants :

- versements de flux applicatifs, afin de les automatiser ;
- versements de dossiers ou de documents dits « sériels », obéissant strictement à des règles de nommage et de description uniformes (par exemple, des images numérisées par un service d'archives) ;
- versements réguliers et récurrents d'un même type d'archives par différents services producteurs et donc répondant à la volonté de disposer d'une description homogène, y compris pour faciliter les recherches sur celles-ci ;
- versements ponctuels d'archives hétérogènes, quel que soit leur type (bureautiques, audiovisuelles, etc.).

En outre, il peut être utilisé de manière différente en permettant de :

- faire appel uniquement à ses services back-office et ses API, automatiser des flux de versements émanant de systèmes d'information ;
- construire des interfaces, appelant ses API permettant de générer un SIP, en vue de faciliter la saisie d'un bordereau de transfert et la préparation d'un versement ;
- utiliser les interfaces de l'APP « Collecte et préparation des versements » du front-office VitamUI.

3. Mécanismes mis en œuvre dans la solution logicielle Vitam

La solution logicielle Vitam offre à un service d'archives plusieurs fonctionnalités lui permettant de **collecter des archives** au moyen du module de collecte :

- la **configuration** des (pré-)versements (ou transactions) au moyen de la définition d'un projet de versement ;
- leur **(pré-)versement** (ou collecte) dans le module ;
- leur **recherche** et leur consultation ;
- leur **gestion** et leur traitement ;
- leur **transfert** vers les espaces de conservation pérennes de la solution logicielle Vitam.

3.1. Configuration

3.1.1. Définitions

Dans la solution logicielle Vitam, il est possible de configurer le versement de un à plusieurs SIP conforme(s) au Standard d'échanges des données pour l'archivage (SEDA) en initialisant un projet de versement au sein du module de collecte.

Un projet de versement est propre à chaque tenant de la solution logicielle Vitam.

Il permet de préconfigurer dans une base de données dédiée, sous forme d'enregistrements JSON, les informations relatives à :

- des métadonnées correspondant à l'en-tête du message ArchiveTransfer (Base Collect > Collection Project),
- le rattachement automatisé à une position dans le tenant de destination,
- les métadonnées descriptives et de gestion attendues dans le cas d'un versement en lot d'archives (*service non implémenté*),
- la fréquence des transferts vers la solution logicielle Vitam, etc. (*service non implémenté*).

Les fonctionnalités ont été conçues et réalisées de manière à prendre en compte 1 à n transaction(s) pour un projet de versement.

Un projet de versement est modifiable.

Points d'attention :

- La création et la modification d'un projet de versement dans le module de collecte ne sont pas journalisées dans le journal des opérations.
- Il est recommandé d'avoir une seule transaction en cours d'alimentation pour un projet de versement.

3.1.2. Utilisation des API

L'envoi et l'enregistrement d'un projet de versement dans le module de collecte s'effectue au moyen d'une commande de création d'un projet de versement correspondant aux métadonnées d'en-tête.

Point d'attention : la création d'un projet de versement est un prérequis à la création des transactions (ou versements).

Lors de la création d'un projet de versement sont envoyées :

- les métadonnées correspondant aux informations dites d'en-tête du bordereau telles que l'identifiant du message ou le contrat d'entrée ;
- le cas échéant, la position .

Un projet de versement peut comporter les éléments suivants[1] :

Champ	Description
Name	Intitulé du projet de versement (champ facultatif)[2].
ArchivalAgreement	Identifiant du contrat d'entrée utilisé pour réaliser l'entrée, destiné à alimenter le champ ArchivalAgreement du message ArchiveTransfer (champ facultatif).
MessageIdentifier	Identifiant du lot d'objets, utilisé pour identifier les versements, destiné à alimenter le champ MessageIdentifier du message ArchiveTransfer (champ facultatif).
Comment	Précisions sur la demande de transfert destiné à alimenter le champ Comment du message ArchiveTransfer (champ facultatif).
OriginatingAgencyIdentifier	Identifiant du service producteur, destiné à alimenter le champ OriginatingAgencyIdentifier du message ArchiveTransfer (champ facultatif).
SubmissionAgencyIdentifier	Identifiant du service versant, destiné à alimenter le champ SubmissionAgencyIdentifier du message ArchiveTransfer (champ facultatif).
ArchivalAgencyIdentifier	Identifiant du service d'archivage, destiné à alimenter le champ ArchivalAgencyIdentifier du message ArchiveTransfer (champ facultatif).
TransferringAgencyIdentifier	Identifiant du service de transfert, destiné à alimenter le champ TransferringAgencyIdentifier du message ArchiveTransfer (champ facultatif).

1 Pour plus d'informations, consulter le document *Modèle de données*, « Collection Project ». Un exemple de projet de versement se trouve dans l'annexe 1 du présent document.

2 À noter que, dans l'APP « Collecte et préparation des versements », ce champ est alimenté par la valeur du champ « MessageIdentifier ».

ArchiveProfile	Identifiant du profil d'archivage utilisé pour réaliser l'entrée, destiné à alimenter le champ ArchivalProfile du message ArchiveTransfer (champ facultatif).
UnitUp	Identifiant de l'unité archivistique à laquelle rattacher automatiquement la ou les unités racines des transactions associées au projet de versement (champ facultatif).
AcquisitionInformation	Modalité d'entrée. Champ destiné à alimenter le champ AcquisitionInformation du message ArchiveTransfer (champ facultatif).
LegalStatus	Statut légal des archives, destiné à alimenter le champ LegalStatus du message ArchiveTransfer (champ facultatif). Si le champ est renseigné, les valeurs attendues sont : « Public Archive », « Private Archive », « Public and Private Archive ».

Points d'attention : Au terme de la version 6.RC :

- on ne peut pas inclure de règles de gestion dans un projet de versement ;
- aucun contrôle n'est effectué avec les référentiels et unités archivistiques de rattachement présents dans la solution logicielle Vitam. De fait, il est fortement recommandé de :
  - veiller à inclure des références (ex. service producteur, contrat d'entrée, etc.) existant dans la solution, de manière à éviter de possibles échecs lors des étapes de contrôles des données référentielles du processus d'entrée (opération « INGEST ») ;
  - ne pas envoyer dans le module de collecte des éléments sans valeurs, en particulier ceux qui sont obligatoires et doivent nécessairement être renseignés.
- la plupart des champs d'un projet de versement sont facultatifs. Néanmoins, il est recommandé d'indiquer un intitulé (Name) et les éléments obligatoires à la génération d'un SIP :
  - le contrat d'entrée (champ « ArchivalAgreement »),
  - l'identifiant du message (champ « MessageIdentifier),
  - le service producteur (champ « OriginatingAgencyIdentifier »),
  - le service d'archivage (champ « ArchivalAgencyIdentifier »),
  - le service responsable du transfert (« TransferringAgencyIdentifier »)

Exemple : requête de création d'un projet de versement

```
POST {{url}}/collect-external/v1/projects
accept: application/json
content-type: application/json
Content-Length: 401
X-Tenant-Id: {{tenant}}

{
  "Name": "Versements du SG",
  "ArchivalAgencyIdentifier": "Vitam",
```

```
  "TransferringAgencyIdentifier": "RATP",
  "OriginatingAgencyIdentifier": "RATP",
  "SubmissionAgencyIdentifier": "RATP",
  "MessageIdentifier": "20200131-000013",
  "ArchivalAgreement":"IC-000001",
  "Comment": "SG – bureautique",
   "UnitUp": "aeaqaaaaaehedmpfaay5gambwxsspviaaaba"
}
```

Cette action provoque :

- la création d'un enregistrement dans la base de données MongoDB, dans la collection « Project » (base *Collect*) ;
- l'ajout des informations suivantes, associées à cet enregistrement :
  - un statut (champ « Status ») dont la valeur est égale à « OPEN »,
  - une date de création (champ « CreationDate ») dont la valeur est égale à la date de création de l'enregistrement,
  - une date de dernière modification (champ « LastUpdate ») dont la valeur est égale à la date de création de l'enregistrement.

Exemple : enregistrement du projet de versement dans la collection « Project »

```
{
   _id: 'aeeaaaaaaghj3m7nabjocamcdqvqqviaaaaq',
   Name: 'Versements du SG',
   Context: {
      ArchivalAgreement: 'IC-000001',
      MessageIdentifier: '20200131-000013',
      ArchivalAgencyIdentifier: 'Vitam',
      TransferirngAgencyIdentifier: 'RATP',
      OriginatingAgencyIdentifier: 'RATP',
      SubmissionAgencyIdentifier: 'RATP',
      Comment: 'SG - bureautique',
      UnitUp: 'aeaqaaaaaehedmpfaay5gambwxsspviaaaba'
   },
      CreationDate: '2022-08-23T09:05:52.242',
      LastUpdate: '2022-08-23T09:05:52.242',
      Status: 'OPEN',
   _tenant: 1
}
```

Lors de cette action, l'opération peut aboutir aux résultats suivants :

Statut	Motifs
Succès	Action réalisée sans rencontrer de problèmes particuliers.
Échec	Présence d'un champ inconnu dans la requête.

3.1.3. Utilisation dans VitamUI

L'APP « Collecte et préparation des versements » du front-office VitamUI fournie avec la solution logicielle Vitam permet de créer un projet de versement au moyen d'un wizard.
Ce projet de versement donne lieu au renseignement :

- d'un possible rattachement à un nœud d'arbre de positionnement ou de plan de classement,
- d'informations décrivant le versement (intitulé du message, description, service producteur et service versant),
- d'informations contextuelles (service d'archives et service responsable du transfert des archives, contrat d'entrée, profil d'archivage, modalité d'entrée, statut légal des versements).

La création du projet dans l'APP entraîne la création d'une unique transaction.

Le projet est par ailleurs accessible dans la liste des projets de versements présente dans l'APP « Collecte et préparation des versements ».

Points d'attention : Au terme de la version 6.RC :

- l'intitulé saisi dans le wizard incrémente à la fois l'identifiant du message (MessageIdentifier) et l'intitulé du projet (Name) ;
- on ne peut pas inclure de règles de gestion dans un projet de versement ;
- aucun contrôle n'est effectué avec les référentiels présents dans la solution logicielle Vitam. De fait, il est fortement recommandé de :
  - veiller à inclure des références (ex. service producteur, contrat d'entrée, etc.) existant dans le tenant (coffre), de manière à éviter de possibles échecs lors des étapes de contrôles des données référentielles du processus d'entrée (opération « INGEST ») ;
  - ne pas envoyer dans le module de collecte des éléments sans valeurs, en particulier ceux qui sont obligatoires et doivent nécessairement être renseignés.
- il n'est pas possible de modifier un projet de versement depuis les interfaces. De fait, en cas d'erreur, il est recommandé de veiller à ne pas faire d'erreurs lors de la saisie des informations.
- l'usage du champ « Profil d'archivage » est déconseillé pour l'instant, car la valeur renseignée depuis les interfaces n'est pas enregistrée dans la base de données.

3.2. Versement

3.2.1. Définitions

Dans la solution logicielle Vitam, il est possible de préparer le versement d'un SIP conforme au Standard d'échanges des données pour l'archivage (SEDA) au moyen du module de collecte.

Ce module est propre à chaque tenant de la solution logicielle Vitam.

Il permet de recevoir dans un espace de stockage et des bases de données dédiés :

- sous forme d'enregistrements JSON les informations relatives à :
  - des métadonnées correspondant à l'en-tête du message ArchiveTransfer (Base Collect > Collection Transaction),
  - des métadonnées descriptives et de gestion associées à des unités archivistiques (Base MetadataCollect > Collection Unit),
  - le cas échéant, des métadonnées techniques (Base MetadataCollect > Collection ObjectGroup),
- les objets associés.

À l'étape d'ajout des objets, la solution logicielle Vitam réalise une mise à jour des métadonnées techniques, qui inclut :

- un calcul d'empreinte de l'objet,
- un calcul du poids de l'objet (exprimé en octets),
- une identification de format.

L'envoi des métadonnées et des objets peut s'effectuer de deux manières :

- en mode « unitaire » : pour chaque type de métadonnées et pour chaque objet à collecter, le module de collecte reçoit une requête d'envoi ;
- en mode « lot » : pour un ensemble d'unités archivistiques et d'objets, le module de collecte reçoit une unique requête lui faisant parvenir l'ensemble.

Les fonctionnalités ont été conçues et réalisées de manière à prendre en compte :

- 1 à n unités archivistiques pour une transaction, correspondant à un ensemble de métadonnées d'en-tête,
- 0 à n parents pour les unités archivistiques,
- 1 groupe d'objets techniques par unité archivistique,
- 1 objet binaire pour une version d'usage de l'objet pour un groupe d'objets techniques donné.

Point d'attention : Le versement des métadonnées et des objets dans le module de collecte n'est pas journalisé dans le journal des opérations.

Après réception, le module a vocation à formater ces différentes informations sous la forme de SIP conformes au standard SEDA, puis à les transférer vers le back-office de la solution logicielle Vitam (opération de type « INGEST »). Cette dernière opération est journalisée dans le journal des opérations (« INGEST »).

3.2.2. Utilisation des API pour envoi d'une transaction

À un projet de versement peu(ven)t être associée(s) 1 à n transaction(s) (ou versement(s)).

Dans la transaction sont envoyées les métadonnées correspondant aux informations dites d'en-tête du bordereau telles que l'identifiant du message ou le contrat d'entrée.

Points d'attention :

- En prérequis à la création d'une transaction, il faut avoir au préalable créé un projet de versement et le signaler dans l'API.
- La création d'une transaction est un prérequis à l'ajout d'archives en mode « unitaire » ou en mode « lot ».

La transaction peut comporter les éléments suivants[3] :

Champ	Description
ArchivalAgreement	Identifiant du contrat d'entrée utilisé pour réaliser l'entrée, destiné à alimenter le champ ArchivalAgreement du message ArchiveTransfer (champ facultatif).
MessageIdentifier	Identifiant du lot d'objets, utilisé pour identifier les versements, destiné à alimenter le champ MessageIdentifier du message ArchiveTransfer (champ facultatif).
Comment	Précisions sur la demande de transfert destiné à alimenter le champ Comment du message ArchiveTransfer (champ facultatif).
OriginatingAgencyIdentifier	Identifiant du service producteur, destiné à alimenter le champ OriginatingAgencyIdentifier du message ArchiveTransfer (champ facultatif).
SubmissionAgencyIdentifier	Identifiant du service versant, destiné à alimenter le champ SubmissionAgencyIdentifier du message ArchiveTransfer (champ facultatif). S'il est absent ou vide à l'initialisation de la transaction, alors la valeur contenue dans le champ OriginatingAgencyIdentifier est reportée dans ce champ.
ArchivalAgencyIdentifier	Identifiant du service d'archivage, destiné à alimenter le champ ArchivalAgencyIdentifier du message ArchiveTransfer (champ facultatif).
TransferringAgencyIdentifier	Identifiant du service de transfert, destiné à alimenter le champ TransferringAgencyIdentifier du message ArchiveTransfer (champ facultatif).
ArchiveProfile	Identifiant du profil d'archivage utilisé pour réaliser l'entrée, destiné à alimenter le champ ArchivalProfile du message ArchiveTransfer (champ facultatif).
AcquisitionInformation	Modalité d'entrée. Champ destiné à alimenter le champ AcquisitionInformation du message ArchiveTransfer (champ facultatif).

3 Pour plus d'informations, consulter le document *Modèle de données*, « Collection Transaction ». Un exemple de contexte de collecte se trouve dans l'annexe 1 du présent document.

LegalStatus	Statut légal des archives, destiné à alimenter le champ LegalStatus du message ArchiveTransfer (champ facultatif). Si le champ est renseigné, les valeurs attendues sont : « Public Archive », « Private Archive », « Public and Private Archive ».

Points d'attention : Au terme de la version 6.RC :

- on ne peut pas inclure de règles de gestion dans la partie transactionnelle ;
- aucun contrôle n'est effectué avec les référentiels et unités archivistiques de rattachement présents dans la solution logicielle Vitam. De fait, il est fortement recommandé de :
  - veiller à inclure des références (ex. service producteur, contrat d'entrée, etc.) existant dans la solution, de manière à éviter de possibles échecs lors des étapes de contrôles des données référentielles du processus d'entrée (opération « INGEST ») ;
  - ne pas envoyer dans le module de collecte des éléments sans valeurs, en particulier ceux qui sont obligatoires et doivent nécessairement être renseignés ;
  - renseigner les valeurs attendues par le SEDA pour le statut légal des archives, à savoir : « Public Archive », « PrivateArchive » ou « Public and Private Archive ».
- la plupart des champs d'une transaction sont facultatifs. Néanmoins, il est recommandé d'indiquer les éléments obligatoires à la génération d'un SIP :
  - le contrat d'entrée (champ « ArchivalAgreement »),
  - l'identifiant du message (champ « MessageIdentifier),
  - le service producteur (champ « OriginatingAgencyIdentifier »),
  - le service d'archivage (champ « ArchivalAgencyIdentifier »),
  - le service responsable du transfert (« TransferringAgencyIdentifier »)

Exemple : requête de création d'une transaction pour le projet de versement préalablement créé dont l'identifiant est aeeaaaaaaghiyso4ablmyal74slqwtqaaaaq.

```
@project-id = aeeaaaaaaghiyso4ablmyal74slqwtqaaaaq
@tenant = 1

POST {{url}}/collect-external/v1/projects/{{project-id}}/transactions
Accept: application/json
Content-Type: application/json
X-Tenant-Id: {{tenant}}

{
 "ArchivalAgencyIdentifier": "Vitam",
 "TransferringAgencyIdentifier": "AN",
 "OriginatingAgencyIdentifier": "MICHEL_MERCIER",
 "SubmissionAgencyIdentifier": "MICHEL_MERCIER",
 "MessageIdentifier": "20220302-000005",
 "ArchivalAgreement":"IC-000001",
 "Comment": "Versement du service producteur : Cabinet de Michel Mercier"
}
```

Cette action provoque la création d'un enregistrement dans la base de données MongoDB, dans la collection « Transaction »[4] (base *Collect*).
À cet enregistrement, est associé :

- un statut (champ « Status ») dont la valeur est égale à « OPEN »,
- l'identifiant du projet de versement à l'origine de la création de la transaction (champ « ProjectId »),
- une date de création (champ « CreationDate ») dont la valeur est égale à la date de création de l'enregistrement,
- une date de dernière modification (champ « LastUpdate ») dont la valeur est égale à la date de création de l'enregistrement.

Exemple : enregistrement de la transaction dans la collection « Transaction »


```
{
   "_id": "aeeaaaaaaghdvam4abgoyambfxnhd6aaaaaq",
   "Status": "OPEN",
   "ProjectId": "aeeaaaaaaghiyso4ablmyal74slqwtqaaaaq",
   "Context": {
      "ArchivalAgreement": "IC-000001",
      "MessageIdentifier": "20220302-000005",
      "ArchivalAgencyIdentifier": "Vitam",
      "TransferringAgencyIdentifier": "AN",
      "OriginatingAgencyIdentifier": "MICHEL_MERCIER",
      "SubmissionAgencyIdentifier": "MICHEL_MERCIER",
      "Comment": "Versement du service producteur : Cabinet de Michel Mercier"
   },
   "CreationDate": "2022-08-23T13:21:41.608",
   "LastUpdate": "2022-08-23T13:21:41.608"
   "_tenant": 1
}
```


Lors de cette action, l'opération peut aboutir aux résultats suivants :

Statut	Motifs
Succès	Action réalisée sans rencontrer de problèmes particuliers.
Échec	Présence d'un champ inconnu dans la requête.

3.2.3. Utilisation des API pour des envois unitaires

L'envoi et l'enregistrement des archives dans le module de collecte s'effectue au moyen de trois commandes distinctes :

4 À noter que cette collection est renommée « Transaction » dans la version 6 RC de la solution logicielle Vitam.

- création unitaire des unités archivistiques,
- création unitaire des groupes d'objets techniques,
- ajout des objets.

Points d'attention :

- En prérequis à la création d'une unité archivistique, il faut avoir au préalable créé une transaction et le signaler dans l'API.
- L'ordre d'utilisation des API doit être respecté en vue d'enregistrer avec succès l'ensemble des métadonnées et objets constituant un versement (ou transaction).

3.2.3.1. Envoi d'unités archivistiques

À une transaction peu(ven)t être associée(s) 1 à n unité(s) archivistique(s).
Chaque unité archivistique :

- doit définir un titre (Title) et un niveau de description (DescriptionLevel), qui sont des éléments rendus obligatoires en entrée de la solution logicielle Vitam ;
- peut inclure :
  - des règles de gestion ;
  - des métadonnées descriptives supplémentaires ;
  - un lien vers une unité archivistique de niveau supérieur.

Points d'attention :

- En prérequis à la création d'une unité archivistique, il faut avoir au préalable créé une transaction et la signaler dans l'API. En outre, la transaction associée doit avoir un statut « OPEN »[5] ;
- Au terme de la version 6 RC, aucun contrôle n'est effectué avec les référentiels présents dans la solution logicielle Vitam. De fait, il est fortement recommandé de veiller à :
  - inclure les métadonnées obligatoires attendues par la solution logicielle Vitam, à savoir un titre (Title) et un niveau de description (DescriptionLevel) ;
  - inclure des règles de gestion ou des identifiants de profil d'unité archivistique existant dans la solution, de manière à éviter de possibles échecs lors des étapes de contrôles des données référentielles du processus d'entrée (opération « INGEST ») ;
  - veiller au bon nommage des éléments (ou vocabulaires internes), tout en respectant leurs caractéristiques (s'il s'agit d'une chaîne de caractères, d'une date, etc.), et à leur insertion dans l'ontologie, s'il s'agit de vocabulaires externes ;
  - ne pas envoyer dans le module de collecte des éléments sans valeurs, en particulier ceux qui sont obligatoires et doivent nécessairement être renseignés.

Exemple : requête de création d'une unité archivistique pour la transaction préalablement créée dont l'identifiant est aeeaaaaaaghiyso4ablmyal74slqwtqaaaaq.

5 Si la transaction est clôturée (son statut est égal à « READY »), il n'est plus possible de lui adjoindre des unités archivistiques, ainsi que des métadonnées techniques et des objets.

```
@transaction-id = aeeaaaaaaghiyso4ablmyal74slqwtqaaaaq
@tenant = 1

POST  {{url}}/collect-external/v1/transactions/{{transaction-id}}/units
Accept: application/json
Content-Type: application/json
X-Tenant-Id: {{tenant}}
X-Access-Contract-Id: ContratTNR
{
   "DescriptionLevel": "RecordGrp",
   "Title": "Discours du ministre",
   "#management": {
     "AccessRule": {
         "Rules": [
            {
               "Rule": "ACC-00001",
               "StartDate": "2000-01-01"
            }
         ]
      }
    }
}
```

Points d'attention :

- L'élément **#management** doit **toujours** être présent dans la requête, même si aucune règle de gestion n'est définie pour une unité archivistique donnée ;

Exemple : requête de création d'une unité archivistique qui ne définit pas de règle de gestion.

```
POST  {{url}}/collect-external/v1/transactions/{{transaction-id}}/units
Accept: application/json
Content-Type: application/json
X-Tenant-Id: {{tenant}}
X-Access-Contract-Id: ContratTNR

{
   "DescriptionLevel": "RecordGrp",
   "Title": "Discours de Michel Mercier",
   "#management": {
   }
}
```

- Pour **associer une unité archivistique à une autre unité archivistique**, de niveau supérieur, il est recommandé de :

- créer en premier lieu l'unité archivistique de niveau supérieur et de récupérer son identifiant technique,
- ajouter parmi les métadonnées de l'unité archivistique de niveau inférieur le paramètre #unitups avec pour valeur l'identifiant de(s) unité(s) archivistique(s) de niveau supérieur.

Exemple : requête de création d'une unité archivistique incluant un rattachement à une unité archivistique de niveau supérieur.

```
POST  {{url}}/collect-external/v1/transactions/{{transaction-id}}/units
Accept: application/json
Content-Type: application/json
X-Tenant-Id: {{tenant}}
X-Access-Contract-Id: ContratTNR

{
   "DescriptionLevel": "Item",
   "Title": "Discours prononcé à l'occasion de l'audition devant la commission des lois relative à la proposition de loi constitutionnelle relative à la fonction de représentation par le Sénat des collectivités locales",
   "TransactedDate":"2010-12-08",
   "#unitups": ["aeeaaaaaaghiyso4ablmyal74smvqcqaaaaq"],
   "#management": {
   }
}
```

Cette action provoque la création d'un enregistrement dans la base de données MongoDB, dans la collection « Unit » (base *MetadataCollect*)[6].
À cet enregistrement est associé l'identifiant de la transaction (_opi).

Exemple : enregistrement de l'unité archivistique dans la collection « Unit » de la base « MetadataCollect ».


```
{
   "_id": "aeeaaaaaaghdvam4abgoyambfxvoaoiaaaaq",
   "DescriptionLevel": "RecordGrp",
   "Title": "Discours du ministre",
   "_mgt": {
      "AccessRule": {
         "Rules": [
            {
               "Rule": "ACC-00001",
               "StartDate": "2000-01-01"
            }
         ]
      }
   },
   "_opi": "aeeaaaaaaghdvam4abgoyambfxnhd6aaaaaq",
```


6 Pour plus d'informations sur l'enregistrement de l'unité archivistique dans la base de données, il est recommandé de consulter le document *Modèle de données*.

```
  "_unitType": "INGEST",
  "_up": [],
  "_us": [],
  "_graph": [],
  "_sps": [],
  "_uds": {},
  "_min": 1,
  "_max": 1,
  "_glpd": "2022-06-04T08:52:56.689",
  "_acd": "2022-06-04T08:52:56.689",
  "_aud": "2022-06-04T08:52:56.689",
  "_v": 0,
  "_av": 0,
  "_tenant": 1
}
```


Lors de cette action, l'opération peut aboutir aux résultats suivants :

Statut	Motifs
Succès	Action réalisée sans rencontrer de problèmes particuliers.
Échec	- L'unité archivistique n'a pas été associée à une transaction ; - La transaction associée à l'unité archivistique n'existe pas ; - La transaction associée à l'unité archivistique a été clôturée.

3.2.3.2. Envoi des métadonnées techniques

À une unité archivistique donnée peut être associé un groupe d'objets techniques, dans lequel on peut ajouter unitairement une version d'objet d'un usage particulier.
Il est possible d'associer plusieurs usages à un groupe d'objets techniques :

- original numérique (BinaryMaster),
- original physique (PhysicalMaster),
- copie numérique (Dissemination),
- vignette (Thumbnail),
- texte brut (TextContent).

Points d'attention :

- En prérequis à la création de métadonnées techniques, il faut avoir au préalable créé :
  - une transaction et la signaler dans l'API. En outre, la transaction associée doit avoir un statut « OPEN »[7] ;
  - une unité archivistique et la signaler dans l'API.

7 Si la transaction est clôturée (son statut est égal à « READY »), il n'est plus possible de lui adjoindre des unités archivistiques, ainsi que des métadonnées techniques et des objets.

- À cette étape, il n'est pas nécessaire d'envoyer les métadonnées correspondant à l'empreinte d'un fichier numérique, à l'identification de son format ou à son poids, dans la mesure où ce type de métadonnées est calculé lors de l'envoi du fichier numérique ;
- Au terme de la version 6 RC, il n'est pas possible de modifier les informations envoyées dans le module de collecte. De fait, il est fortement recommandé de veiller au bon nommage des éléments (ou vocabulaires internes), tout en respectant leurs caractéristiques (s'il s'agit d'une chaîne de caractères, d'une date, etc.), et à leur insertion dans l'ontologie, s'il s'agit de vocabulaires externes.
- Par ailleurs, certains contrôles, disponibles au moment du transfert dans la solution logicielle Vitam n'ont pas été implémentés. Il est fortement recommandé de ne créer dans le module de collecte que des versions initiales d'objet numérique pour un usage donné.

Exemple : requête d'association d'un groupe d'objets techniques, et plus exactement une version initiale d'un original numérique, à une unité archivistique dont l'identifiant est aeeaaaaaaghiyso4ablmyal74sndwiqaaaaq.

```
@transaction-id = aeeaaaaaaghiyso4ablmyal74slqwtqaaaaq
@unit-id = aeeaaaaaaghiyso4ablmyal74sndwiqaaaaq
@tenant = 1

POST  {{url}}/collect-external/v1/units/{{unit-id}}/objects/BinaryMaster/1
Accept: application/json
Content-Type: application/json
X-Tenant-Id: {{tenant}}
X-Access-Contract-Id: ContratTNR

{
  "FileInfo": {
    "Filename": "Test01.jpeg"
  }
}
```

Cette action provoque la création d'un enregistrement dans la base de données MongoDB, dans la collection « ObjectGroup » (base *MetadataCollect*[8]).
À cet enregistrement est associé l'identifiant de la transaction (_opi).

Exemple : enregistrement des métadonnées techniques dans la collection « ObjectGroup » de la base « MetadataCollect ».

```
{
    "_id": "aeeaaaaaaghdvam4abgoyambfxzdbviaaaaq",
    "_tenant": 1,
    "FileInfo": {
```

8 Pour plus d'informations sur l'enregistrement de l'unité archivistique dans la base de données, il est recommandé de consulter le document *Modèle de données*.

```
      "Filename": "Test01.jpeg"
   },
   "_nbc": 2,
   "_opi": "aeeaaaaaaghdvam4abgoyambfxnhd6aaaaaq",
   "_qualifiers": [
      {
         "qualifier": "BinaryMaster",
         "_nbc": 1,
         "versions": [
            {
               "_id": "aeeaaaaaaghdvam4abgoyambfxzdbvyaaaaq",
               "DataObjectVersion": "BinaryMaster_1",
               "FileInfo": {
                  "Filename": "Test01.jpeg"
               },
               "Size": 0
            }
         ]
      }
   ],
   "_v": 1,
   "_av": 1,
   "_acd": "2023-01-02T12:01:34.629",
   "_aud": "2023-01-02T12:01:34.629"
}
```


Lors de cette action, l’opération peut aboutir aux résultats suivants :

Statut	Motifs
Succès	Action réalisée sans rencontrer de problèmes particuliers.
Échec	- L’ unité archivistique n’existe pas ou est erronée ; - L’objet technique n’a pas été associé à une unité archivistique ; - La requête renvoie une deuxième fois une demande de création d’un usage d’objet ayant déjà été versé dans le module de collecte ; - La transaction associée n’existe pas ; - La transaction associée a été clôturée.

3.2.3.3. Envoi des objets

À une unité archivistique associée un groupe d’objets techniques et pour un usage et une version d’objet donné, on peut ajouter unitairement un objet numérique.

Points d’attention :

- En prérequis à l’envoi d’un objet, il faut avoir au préalable créé :

- une transaction et la signaler dans l'API. En outre, la transaction associée doit avoir un statut « OPEN »[9] ;
- une unité archivistique et la signaler dans l'API.

Exemple : requête d'association d'un fichier numérique, et plus exactement une version initiale d'un original numérique, à une unité archivistique dont l'identifiant est aeeaaaaaaghiyso4ablmyal74sndwiqaaaaq.

@transaction-id = aeeaaaaaaghiyso4ablmyal74slqwtqaaaaq
@unit-id = **aeeaaaaaaghiyso4ablmyal74sndwiqaaaaq**
@tenant = 1

POST {{url}}/collect-external/v1/units/{{unit-id}}/objects/**BinaryMaster/1/binary**
Accept: application/json
Content-Type: application/octet-stream
X-Tenant-Id: {{tenant}}
X-Access-Contract-Id: ContratTNR

< MichelMercier/Discours/ Test01.jpeg

Cette action provoque :

- l'enregistrement de l'objet numérique sur les offres de stockage.
- la mise à jour des métadonnées techniques de l'objet avec, calculés lors de l'envoi de l'objet numérique :
  - ajout de l'empreinte d'un fichier numérique,
  - ajout de l'identification de son format,
  - mise à jour de son poids exprimé en octets.

Exemple : ajout de l'empreinte du fichier et de son identification de format et mise à jour du poids dans la collection « ObjectGroup » de la base « MetadataCollect ».


```
{
   "_id": "aeeaaaaaaghdvam4abgoyambfxzdbviaaaaq",
   "_tenant": 1,
   "FileInfo": {
      "Filename": "Test01.jpeg"
   },
   "_nbc": 2,
   "_opi": "aeeaaaaaaghdvam4abgoyambfxnhd6aaaaaq",
   "_qualifiers": [
      {
         "qualifier": "BinaryMaster",
         "_nbc": 1,
         "versions": [
            {
```


9 Si la transaction est clôturée (son statut est égal à « READY »), il n'est plus possible de lui adjoindre des unités archivistiques, ainsi que des métadonnées techniques et des objets.

```
                "_id": "aeeaaaaaaghdvam4abgoyambfxzdbvyaaaaq",
                "DataObjectVersion": "BinaryMaster_1",
                "FormatIdentification": {
                   "FormatLitteral": "JPEG File Interchange Format",
                   "MimeType": "image/jpeg",
                   "FormatId": "fmt/43"
                },
                "FileInfo": {
                   "Filename": "Test01.pdf"
                },
                "Size": 7702,
                "Uri": "Content/Test01jpeg",
                "MessageDigest":
"0e0cec05a1d72ee5610eaa5afbc904c012d190037cbc827d08272102cdecf0226efcad122b86e7699f767c661c9f3702
379b8c2cb01c4f492f69deb200661bb9",
                "Algorithm": "SHA-512"
             }
          ]
       }
   ],
   "_v": 1,
   "_av": 1,
   "_acd": "'023-01-02T12:01:34.629",
   "_aud": "2023-01-02T12:01:34.965"
}
```


Lors de cette action, l'opération peut aboutir aux résultats suivants :

Statut	Motifs
Succès	Action réalisée sans rencontrer de problèmes particuliers.
Échec	- L' unité archivistique n'existe pas ou est erronée ; - La transaction associée n'existe pas ; - La transaction associée a été clôturée.

Point d'attention : Certains contrôles, disponibles au moment du transfert dans la solution logicielle Vitam, n'ont pas été implémentés. Il est fortement recommandé de ne pas envoyer dans le module de collecte plusieurs objets numériques pour un usage et une version donnée.

3.2.4. Utilisation des API pour des envois en lot

L'envoi et l'enregistrement des données dans le module de collecte peuvent s'effectuer en lot au moyen d'une commande fonctionnant de deux façons :

- envoi sous forme de zip d'une arborescence bureautique,
- envoi sous forme de zip d'une arborescence bureautique et d'un fichier annexe intitulé « metadata.csv » référençant des métadonnées descriptives et de gestion.

Point d'attention : Pour ces deux envois, la commande est unique. Seul l'association d'un fichier annexe « metadata.csv », optionnelle, diffère.

3.2.4.1. Envoi d'une arborescence bureautique

Pour une transaction donnée est envoyée une arborescence bureautique sous forme de zip.

Point d'attention :

- En prérequis à l'envoi d'une arborescence bureautique, il faut avoir au préalable créé une transaction et le signaler dans l'API.

Exemple : requête d'envoi d'une arborescence bureautique stream.zip pour la transaction préalablement créée dont l'identifiant est aeeaaaaaaghiyso4ablmyal74slqwtqaaaaq.

@transaction-id = **aeeaaaaaaghiyso4ablmyal74slqwtqaaaaq**
@tenant = 1

POST {{url}}/collect-external/v1/transactions/{{transaction-id}}/upload
Accept: application/json
Content-Type: application/zip
X-Tenant-Id: {{tenant}}
X-Access-Contract-Id: {{access-contract}}

< /path_tozip/**stream.zip**

Cette action provoque :

- la création des unités archivistiques dans la base de données MongoDB, dans la collection « Unit » (base *MetadataCollect*[10]). Sont enregistrés automatiquement[11] :
  - un niveau de description (DescriptionLevel) dont la valeur est :
    - « RecordGrp » pour une unité archivistique référençant un répertoire,
    - « Item » pour une unité archivistique référençant un objet numérique ;
  - un intitulé (Title), correspondant au nom d'un répertoire ou d'un objet binaire présent dans l'arborescence bureautique.

  À chaque enregistrement est associé l'identifiant de la transaction (_opi).
- la création de métadonnées techniques dans la base de données MongoDB, dans la collection « ObjectGroup » (base *MetadataCollect*[12]) ;
  À chaque enregistrement est associé l'identifiant de la transaction (_opi) ;

10 Pour plus d'informations sur l'enregistrement de l'unité archivistique dans la base de données, il est recommandé de consulter le document *Modèle de données*.

11 Le fonctionnement est identique à celui de l'import d'arborescence bureautique dans ReSIP.

12 Pour plus d'informations sur l'enregistrement de l'unité archivistique dans la base de données, il est recommandé de consulter le document *Modèle de données*.

- l'enregistrement des objets numériques sur les offres de stockage.
- la mise à jour des métadonnées techniques de l'objet avec, calculés lors de l'envoi du fichier numérique :
  - ajout de l'empreinte d'un fichier numérique,
  - ajout de l'identification de son format,
  - mise à jour de son poids exprimé en octets.

Lors de cette action, l'opération peut aboutir aux résultats suivants :

Statut	Motifs
Succès	Action réalisée sans rencontrer de problèmes particuliers.
Échec	Le fichier .zip n'a pas été téléchargé pour cause de nom erroné ou de chemin introuvable La transaction n'existe pas ou est erronée. La transaction a été clôturée.

Elle n'est pas journalisée dans le journal des opérations.

Points d'attention :

- Aucun fichier ne doit avoir un poids équivalent à 0 octet.
- Au terme de la V.6 RC, il est recommandé que les noms de répertoires et de fichiers ne contiennent ni caractère accentué, ni virgule, ni apostrophe, ni parenthèse, ni espace, ni élément de ponctuation, ou tout autre caractère spécial. Ne sont à privilégier que l'underscore et le tiret comme séparateurs.
  Néanmoins, s'ils en contiennent et si l'arborescence bureautique émane d'un environnement Windows, il est recommandé d'utiliser l'outil Winzip pour la zipper, afin d'éviter des problèmes d'encodage.

3.2.4.2. Envoi d'une arborescence bureautique avec fichier .csv de métadonnées

Pour une transaction donnée peut être envoyé sous forme de zip en plus d'une arborescence bureautique[13] un fichier .csv contenant des métadonnées détaillant unitairement les unités archivistiques.

Le fichier .csv, obligatoirement intitulé « metadata.csv », est composé de x colonnes[14] :

- File : chemin relatif à partir de l'emplacement où est enregistré le fichier .csv (colonne obligatoire) ;
- DescriptionLevel : niveau de description de l'unité archivistique (colonne obligatoire) ;
- Title : intitulé de l'unité archivistique (colonne obligatoire) ;

13 Se référer à la sous-section 3.2.4.2 « Envoi d'une arborescence bureautique » du présent document.

14 Son modèle est identique au modèle de fichier .csv importé dans ReSIP, hors colonnes ID et ParentID qui ne sont pas supportées par le module de collecte en l'état actuel des développements.

- toute colonne correspondant à un champ du standard SEDA (colonnes facultatives).

Exemple : contenu d'un fichier « metadata.csv ».

"File";"Content.DescriptionLevel";"Content.Title";"Content.FilePlanPosition";"Content.ArchivalAgencyArchiveUnitIdentifier";"Content.Description";"Content.CustodialHistory.CustodialHistoryItem.0";"Content.CustodialHistory.CustodialHistoryItem.1";"Content.CustodialHistory.CustodialHistoryItem.2";"Content.DocumentType";"Content.Language";"Content.DescriptionLanguage";"Content.Version";"Content.Tag.0";"Content.Tag.1";"Content.Tag.2";"Content.Tag.3";"Content.Tag.4";"Content.Keyword.0.KeywordContent";"Content.Keyword.0.KeywordType";"Content.Keyword.1.KeywordContent";"Content.Keyword.1.KeywordType";"Content.Keyword.2.KeywordContent";"Content.Keyword.2.KeywordType";"Content.Keyword.3.KeywordContent";"Content.Keyword.3.KeywordType";"Content.Keyword.4.KeywordContent";"Content.Keyword.4.KeywordType";"Content.Coverage.Spatial.0";"Content.Coverage.Spatial.1";"Content.Coverage.Spatial.2";"Content.Coverage.Spatial.3";"Content.Coverage.Temporal";"Content.Coverage.Juridictional.0";"Content.Coverage.Juridictional.1";"Content.Coverage.Juridictional.2";"Content.Coverage.Juridictional.3";"Content.OriginatingAgency.Identifier";"Content.SubmissionAgency.Identifier";"Content.StartDate";"Content.EndDate";"Management.AccessRule.Rule";"Management.AccessRule.StartDate"
"AU1\27juillet1888\5FI6_5FI6_16";"RecordGrp";"Brantes. Combe de la Mure.";"instrument_recherche/5FI6/5FI6_16";"5 Fi 6/16";"Dim. 18x24 cm.";;;;;"fre";"fre";"Original";"Combe de la Mure (Brantes, Vaucluse, France)";"Brantes (Vaucluse, France)";;;;"Combe de la Mure (Brantes, Vaucluse, France)";"geogname";"Brantes (Vaucluse, France)";"geogname";;;;;;;"Combe de la Mure (Brantes, Vaucluse, France)";"Brantes (Vaucluse, France)";;;1888;"Combe de la Mure (Brantes, Vaucluse, France)";"Brantes (Vaucluse, France)";;;;;;"1888-12-30T00:00:00";"ACC-00001";"1888-12-30"
"AU1\27juillet1888";"RecordGrp";"Clichés du 27 juillet 1888";;;;;;;;"fre";"fre";;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

<u>Points d'attention :</u>

- l'ordre des premières colonnes ne doit pas être modifié ;
- une première ligne d'en-tête donnant le nom des colonnes doit être présente, chaque ligne décrivant ensuite une unité archivistique ;
- le séparateur entre les colonnes est le point-virgule, le séparateur de texte les guillemets doubles et l'encodage est « UTF-8 » ;
- le fichier .csv ne référence que des métadonnées propres aux unités archivistiques (métadonnées descriptives et de gestion).
- seul un objet peut être associé à un enregistrement. Ce format d'import ne permet pas de facto de gérer l'import de groupe d'objets techniques disposant de plusieurs objets aux usages différents devant être référencés par la même unité archivistique.
- le fichier .csv à importer doit se trouver dans le même répertoire que le répertoire correspondant à la racine de la structure arborescente de fichiers à importer.
- chaque répertoire et objet numérique devant contenir des métadonnées particulières doit être référencé dans le fichier .csv ;
- concernant le nommage des colonnes :
  - pour les colonnes correspondant à des champs du standard SEDA, l'intitulé de la colonne doit correspondre à celui du champ dans le standard SEDA, précédé de « Management. » s'il s'agit d'une métadonnée de gestion (ex. « Management.AccessRule.Rule » pour une règle de communicabilité) ou de « Content » s'il s'agit d'une métadonnée descriptive (ex. « Content.DocumentType ») ;

  - quand le schéma XML du standard SEDA propose une structure complexe de balises (par exemple pour décrire un auteur via l'objet XML <Writer> qui contient plusieurs balises XML comme FullName ou BirthName), il convient d'intituler la colonne de la manière suivante : Content.Writer.FullName ou Content.Writer.BirthName ;
  - quand un champ ou un objet XML est multivalué dans le standard SEDA (et qu'il est possible d'en décrire plusieurs dans le bordereau comme c'est le cas pour l'objet Writer par exemple), il convient de numéroter la colonne de la manière suivante : Content.Writer.0.FullName, Content.Writer.1.FullName ;
- concernant le contenu des colonnes :
  - la colonne File :
    - ne doit pas comprendre d'espace avant ou après les « \ » ;
    - doit correspondre à un chemin tel que décrit par l'explorateur de fichiers (avec des « \ » et non des « / ») ;
  - la colonne DescriptionLevel ne doit comprendre que les valeurs autorisées par le standard SEDA : Collection, Fonds, Series, SubSeries, RecordGrp, File, Item ;
  - les colonnes correspondant à des champs Date dans le standard SEDA doivent être formatées conformément à la norme ISO 8601 (AAAA-MM-JJ) ;
  - les références à des règles de gestion et à des profils d'unité archivistique doivent se conformer aux identifiants de règles présents dans le SAE
- Aucun fichier ne doit avoir un poids équivalent à 0 octet.
- Les dates de début (Content.StartDate) doivent être antérieures aux dates de fin (Content.EndDate).
- Au terme de la V.6 RC, il est recommandé que les noms de répertoires et de fichiers ne contiennent ni caractère accentué, ni virgule, ni apostrophe, ni parenthèse, ni espace, ni élément de ponctuation, ou tout autre caractère spécial. Ne sont à privilégier que l'underscore et le tiret comme séparateurs.
  Néanmoins, s'ils en contiennent et si l'arborescence bureautique émane d'un environnement Windows, il est recommandé d'utiliser l'outil Winzip pour la zipper, afin d'éviter des problèmes d'encodage.

L'import du fichier zip incluant un fichier .csv et une arborescence bureautique provoque :

- la création des unités archivistiques dans la base de données MongoDB, dans la collection « Unit » (base *MetadataCollect*[15]). Sont enregistrées automatiquement les valeurs portées dans le fichier .csv si l'enregistrement ne contient pas d'erreur.
  À chaque enregistrement est associé l'identifiant de la transaction (_opi).
- la création de métadonnées techniques dans la base de données MongoDB, dans la collection « ObjectGroup » (base *MetadataCollect*[16]).
  À chaque enregistrement est associé l'identifiant de la transaction (_opi) ;

15 Pour plus d'informations sur l'enregistrement de l'unité archivistique dans la base de données, il est recommandé de consulter le document *Modèle de données*.

16 Pour plus d'informations sur l'enregistrement de l'unité archivistique dans la base de données, il est recommandé de consulter le document *Modèle de données*.

- l’enregistrement des objets numériques sur les offres de stockage.
- la mise à jour des métadonnées techniques de l’objet avec :
  - ajout de l’empreinte d’un fichier numérique,
  - ajout de l’identification de son format,
  - mise à jour de son poids exprimé en octets, calculés lors de l’envoi du fichier numérique.

Lors de cette action, l’opération peut aboutir aux résultats suivants :

Statut	Motifs
Succès	Action réalisée sans rencontrer de problèmes particuliers.
Échec	Le fichier .zip n’a pas été téléchargé pour cause de nom erroné ou de chemin introuvable La transaction n’existe pas ou est erronée. La transaction est clôturée.

Cette action n’est pas journalisée dans le journal des opérations.

Points d’attention :

- Au terme de la V.6 RC, le module de collecte ne bloque pas l’opération d’import de l’arborescence bureautique accompagnée d’un fichier CSV si ce dernier comporte les erreurs suivantes :
  - il inverse les colonnes Title et DescriptionLevel ;
  - il ne contient pas au moins une colonne obligatoire (File, Title, DescriptionLevel) ;
  - il dispose d’un champ dont le contenu est mal formaté (ex. ReceivedDate écrite en chaîne de caractères) ;
  - il ne contient aucune information ;
  - il contient des virgules, des espaces, des pipes comme séparateurs de champs ;
  - le fichier contient des simples guillemets comme séparateurs de texte ;
  - il contient un champ obligatoire non renseigné (Title) ;
  - il n'est pas au format CSV ;
  - il ne contient pas de séparateurs de champs.

  Si le fichier CSV est erroné, une erreur peut être renvoyée par l’API, le contenu du fichier CSV sera ignoré et seule l’arborescence bureautique sera téléchargée selon le comportement décrit dans la sous-section précédente.
- Aucun contrôle n’est effectué entre le nombre de répertoires et d’objets binaires présents dans l’arborescence bureautique et les éléments décrits dans le fichier .csv. Il est recommandé de veiller à ne pas ajouter de niveaux intermédiaires dans l’arborescence bureautique non référencés dans le fichier .csv, car ils seront automatiquement créés dans le module de collecte selon le comportement décrit dans la sous-section précédente.

3.2.5. Utilisation dans VitamUI

L'APP « Collecte et préparation des versements » du front-office VitamUI fournie avec la solution logicielle Vitam permet de verser une arborescence bureautique, le cas échéant accompagnée d'un fichier « metatadata.csv », lors de la création d'un projet de versement au moyen d'un wizard.

Le détail du projet de versement, ainsi que les archives qui lui sont associées sont par ailleurs accessibles depuis l'APP.

Points d'attention :

- Si un fichier metadata.csv est associé à une arborescence bureautique, il faut d'abord télécharger cette dernière dans le wizard, puis faire de même avec le fichier « metadata.csv » ;
- Si le nom du fichier .csv est erroné, l'interface VitamUI l'interprètera comme un fichier numérique à collecter au même titre que l'arborescence bureautique associée ;
- Au terme de la V.6 RC, l'APP « Collecte et préparation des versements » ne bloque pas l'opération d'import de l'arborescence bureautique accompagnée d'un fichier CSV si ce dernier comporte les erreurs suivantes :
  - il inverse les colonnes Title et DescriptionLevel ;
  - il ne contient pas au moins une colonne obligatoire (File, Title, DescriptionLevel) ;
  - il dispose d'un champ dont le contenu est mal formaté (ex. ReceivedDate écrite en chaîne de caractères) ;
  - il ne contient aucune information ;
  - il contient des virgules, des espaces, des pipes comme séparateurs de champs ;
  - le fichier contient des simples guillemets comme séparateurs de texte ;
  - il contient un champ obligatoire non renseigné (Title) ;
  - il n'est pas au format CSV ;
  - il ne contient pas de séparateurs de champs.

  Si le fichier CSV est erroné, une erreur peut être renvoyée, le contenu du fichier CSV sera ignoré et seule l'arborescence bureautique sera téléchargée.
- Aucun contrôle n'est effectué entre le nombre de répertoires et d'objets binaires présents dans l'arborescence bureautique et les éléments décrits dans le fichier .csv.
- Au terme de la V.6 RC, il est recommandé que les noms de répertoires et de fichiers ne contiennent ni caractère accentué, ni virgule, ni apostrophe, ni parenthèse, ni espace, ni élément de ponctuation, ou tout autre caractère spécial. Ne sont à privilégier que l'underscore et le tiret comme séparateurs.

3.3. Accès

3.3.1. Définitions

Dans la solution logicielle Vitam, il est possible de rechercher et de consulter les projets de versement, les versements, ainsi que les archives associées, au moyen du module de collecte.

Cette recherche et consultation est propre à chaque tenant de la solution logicielle Vitam.

3.3.2. Utilisation des API

3.3.2.1. Accès aux projets de versement

La solution logicielle Vitam permet de :

- accéder à la liste des projets de versement créés sur un tenant donné,
- consulter les informations d'un projet en particulier sur un tenant donné,
- rechercher un à plusieurs projets de versement par rapport à certains critères (recherche approchante) :
  - un intitulé,
  - un service producteur.

Exemple : requête en vue d'obtenir l'ensemble des projets de versement disponibles sur un tenant donné

@tenant = 1

```
GET {{url}}/collect-external/v1/projects
accept: application/json
X-Tenant-Id: {{tenant}}
```

Exemple : requête en vue d'obtenir les informations relatives à un projet donné dont l'identifiant est « aeeaaaaaaghpmborabjyyambxjekiyyaaaaq » sur un tenant donné.

@project-id = **aeeaaaaaaghpmborabjyyambxjekiyyaaaaq**

```
@tenant = 1

GET {{url}}/collect/v1/projects/{{project-id}}
accept: application/json
content-type: application/json
X-Tenant-Id: {{tenant}}
X-Access-Contract-Id: {{access-contract}}
```

Exemple : requête en vue d'obtenir les projets de versement dont l'intitulé et/ou le service producteur contient le terme « Versement » sur un tenant donné.

```
@tenant = 1

GET {{url}}/collect-external/v1/projects
accept: application/json
content-type: application/json
X-Tenant-Id: {{tenant}}
Content-Length: 143

{
   "$query":"Versement"
}
```

Point d'attention : En résultats, la solution logicielle renvoie systématiquement l'ensemble des métadonnées relatives au(x) projet(s) de versement. Au terme de la version 6.RC, il n'est pas possible de récupérer une sélection de métadonnées.

3.3.2.2. Accès aux transactions

Pour un projet donné, la solution logicielle Vitam permet de :

- accéder à la liste des transactions associées créées sur un tenant donné,
- consulter les informations d'une transaction en particulier sur un tenant donné.

Exemple : requête en vue d'obtenir l'ensemble des transactions d'un projet de versement disponibles sur un tenant donné

```
@tenant = 1
@project-id = aeeaaaaaaghpmborabjyyambxjekiyyaaaaq

GET {{url}}/collect-external/v1/projects/{{project-id}}/transactions
accept: application/json
content-type: application/json
X-Tenant-Id: {{tenant}}
X-Access-Contract-Id: {{access-contract}}
```

Exemple : requête en vue d'obtenir les informations relatives à une transaction donnée dont l'identifiant est « aeeaaaaaaghpmborabjyyambxjekiyyaaabcdef » sur un tenant donné

```
@tenant = 1
@transaction-id = aeeaaaaaaghpmborabjyyambxjekiyyaaabcdef

GET {{url}}/collect-external/v1/transactions/{{transaction-id}}
Accept: application/json
Content-Type: application/json
X-Tenant-Id: {{tenant}}

{
}
```

Point d'attention : En résultats, la solution logicielle renvoie systématiquement l'ensemble des métadonnées relatives au(x) transaction(s). Au terme de la version 6.RC, il n'est pas possible de récupérer une sélection de métadonnées.

3.3.2.3. Accès aux unités archivistiques

L'utilisateur peut rechercher :

- une unité archivistique en particulier et accéder à son détail
- une unité archivistique pour un projet de versement donné,

- une unité archivistique pour une transaction donnée.

Il est possible de :

- limiter le nombre de résultats retournés au moyen d'un seuil de requête ;
- obtenir :
  - une liste de résultats, incluant l'ensemble des métadonnées des unités archivistiques ou une sélection,
  - un résultat par facettes (nombre d'occurrences pour une métadonnée donnée).

Point d'attention : le seuil de résultats supporté par le moteur d'indexation Elastic Search est de 10 000 unités archivistiques. Il est de fait recommandé d'utiliser des requêtes ne dépassant pas les 10 000 résultats.

Exemple : requête en vue d'obtenir l'unité archivistique dont l'identifiant est "aeeaaaaaaceezldyaamscambcvdf6zyaaaaq"

```
@tenant = 1
@unit-id= aeeaaaaaaceezldyaamscambcvdf6zyaaaaq

GET {{url}}/collect-external/v1/units/{{unit-id}}
accept: application/json
content-type: application/json
X-Tenant-Id: {{tenant}}
Content-Length: 143

{

}
```

Exemple : requête en vue d'obtenir les unités archivistiques du projet de versement dont l'identifiant est "aeeaaaaaaceezldyaamscambcvdf6zyaaaaq"

```
@tenant = 1
@project-id= aeeaaaaaaceezldyaamscambcvdf6zyaaaaq

GET {{url}}/collect-external/v1/projects/{{project-id}}/units
accept: application/json
content-type: application/json
X-Tenant-Id: {{tenant}}
Content-Length: 143

{
  "$roots": [],
  "$query": [ {
          "$eq": {
             "#unitType": "INGEST"
          }
       }],
  "$filter": {},
  "$projection": {}
}
```

Exemple : requête en vue d'obtenir les unités archivistiques de la transaction dont l'identifiant est

"aeeaaaaaaceezldyaamscambcvdf6zyaaaaq"

@tenant = 1
@transaction-id= ***aeeaaaaaaceezldyaamscambcvdf6zyaaaaq***

```
GET {{url}}/collect-external/v1/transactions/{{transaction-id}}/units
accept: application/json
content-type: application/json
X-Tenant-Id: {{tenant}}
Content-Length: 143

{
  "$roots": [],
  "$query": [ {
          "$eq": {
             "#unitType": "INGEST"
          }
       }],
  "$filter": {},
  "$projection": {}
}
```

Exemple : requête en vue d'obtenir les unités archivistiques du projet de versement dont l'identifiant est "aeeaaaaaaceezldyaamscambcvdf6zyaaaaq" et ayant pour tout ou partie du champ Title la valeur « Bulletin »

@tenant = 1
@project-id= ***aeeaaaaaaceezldyaamscambcvdf6zyaaaaq***

```
GET {{url}}/collect-external/v1/projects/{{project-id}}/units
accept: application/json
content-type: application/json
X-Tenant-Id: {{tenant}}
Content-Length: 143

{
  "$roots": [],
  "$query": [ { "$match": { "Title": "Bulletins" } } ],
  "$filter": {},
  "$projection": {}
}
```

3.3.2.4. Accès aux groupes d'objets techniques

L'utilisateur peut récupérer :

- les métadonnées techniques d'un groupe d'objets techniques donné,
- l'objet numérique associé à une unité archivistique donnée.

Exemple : requête en vue d'obtenir le groupe d'objets techniques dont l'identifiant est "aeeaaaaaaceezldyaamscambcvdf6zyaaaaq"

```
@tenant = 1
@got-id=  aeeaaaaaaceezldyaamscambcvdf6zyaaaaq

GET {{url}}/collect-external/v1/objects/{{got-id}}
accept: application/json
content-type: application/json
X-Tenant-Id: {{tenant}}
Content-Length: 143

{

}
```

Exemple : requête en vue d'obtenir l'objet technique d'usage « BinaryMaster » et de version « 1 » associé à l'unité archivistique dont l'identifiant est "aeeaaaaaaceezldyaamscambcvdf6zyaaaaq"

```
@tenant = 1
@unit-id=  aeeaaaaaaceezldyaamscambcvdf6zyaaaaq

GET  {{url}}/collect-external/v1/units/{{unit-id}}/objects/BinaryMaster/1/binary
Accept: application/json
Content-Type: application/octet-stream
X-Tenant-Id: {{tenant}}

{

}
```

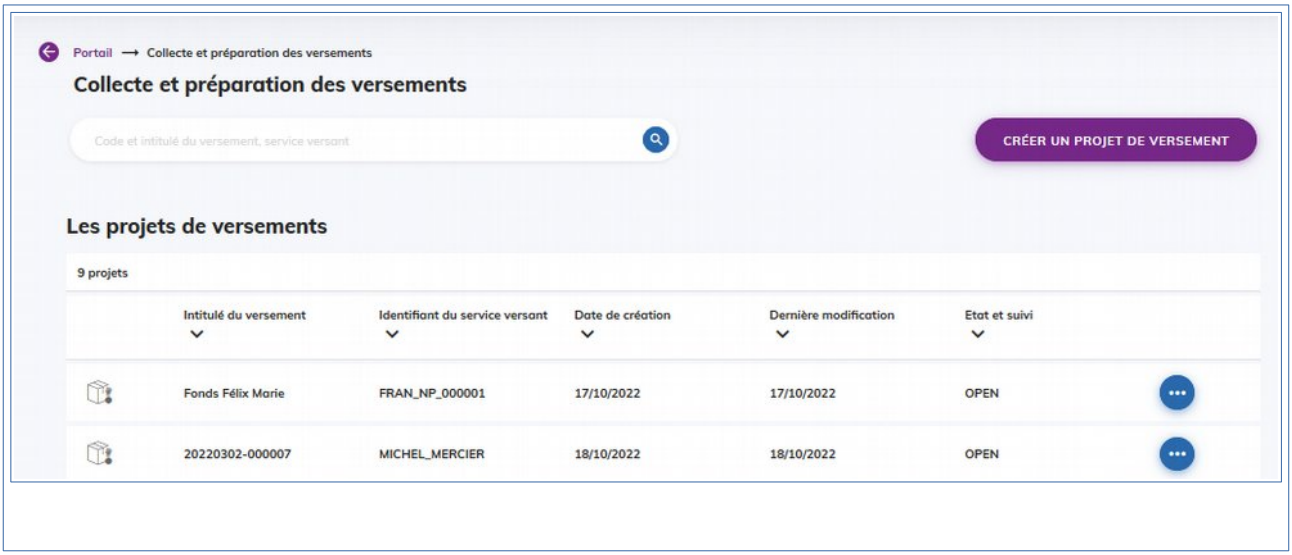
3.3.3. Utilisation dans VitamUI

L'APP « Collecte et préparation des versements » du front-office VitamUI fournie avec la solution logicielle Vitam permet d'accéder à la liste des projets de versement préalablement créés et de les rechercher au moyen de :

- leur intitulé (champ MessageIdentifier),
- leur service versant (champ TransferringAgencyIdentifier).

Point d'attention : La recherche est approchante.

Exemple : champ de recherche dans l'APP « Collecte et préparation des versements »



Par ailleurs, pour un projet donné, il est possible de :

- consulter son détail ;
- consulter les transactions associées ;
- consulter les archives associées :
  - sous forme de liste,
  - via une navigation dans l'arborescence bureautique ;
- les rechercher au moyen de critères de recherche simples ou avancés,
- accéder à leur détail et, le cas échéant, télécharger le fichier numérique associé.

3.4. Gestion des archives

3.4.1. Définitions

Le module de collecte a vocation à permettre d'effectuer un certain nombre de traitement en amont du versement.

Au terme de la V6.RC, il est possible de réaliser les actions suivantes :

- modification,
- suppression,
- abandon,
- réouverture.

Ces actions sont propres à chaque tenant de la solution logicielle Vitam.
Elles ne sont pas journalisées dans le journal des opérations.

3.4.2. Utilisation des API

3.4.2.1. Modification d'un projet de versement

La solution logicielle Vitam permet de modifier un projet de versement, et plus précisément de :

- modifier ses métadonnées,
- lui ajouter des métadonnées,
- supprimer des métadonnées.

Exemple : requête en vue de modifier des métadonnées d'un projet de versement dont l'identifiant est « aeeaaaaaachj3m7nabjocamcdqr2rqaaaaaq »

```
@tenant = 1

PUT {{url}}/collect-external/v1/projects/
accept: application/json
content-type: application/json
Content-Length: 401
X-Tenant-Id: {{tenant}}

{
  "#id": "aeeaaaaaachj3m7nabjocamcdqr2rqaaaaaq",
  "ArchivalAgencyIdentifier": "Vitam",
  "TransferringAgencyIdentifier": "RATP",
  "OriginatingAgencyIdentifier": "RATP",
  "SubmissionAgencyIdentifier": "RATP",
  "MessageIdentifier": "20200131-000006",
  "ArchivalAgreement":"IC-000001",
  "Comment": "Bulletins de salaire - Janvier 2020"
}
```

Points d'attention :

- Le service de mise à jour d'un projet de versement fonctionne en mode « annule et remplace ».
- La plupart des champs d'une transaction étant facultatifs, il est recommandé d'indiquer les éléments obligatoires à la génération d'un SIP :
  - le contrat d'entrée (champ « ArchivalAgreement »),
  - l'identifiant du message (champ « MessageIdentifier),
  - le service producteur (champ « OriginatingAgencyIdentifier »),
  - le service d'archivage (champ « ArchivalAgencyIdentifier »),
  - le service responsable du transfert (« TransferringAgencyIdentifier »)
- Au terme de la version 6.RC :
  - le module de collecte ne permet pas de modifier un rattachement.

- ce service ne vérifie pas le caractère obligatoire d'un champ. Il est fortement recommandé de veiller à ne pas supprimer un champ obligatoire, de manière à éviter de possibles échecs lors des étapes de contrôles des données référentielles du processus d'entrée (opération « INGEST ») ;
- on ne peut pas ajouter de règles de gestion dans un projet de versement ;
- aucun contrôle n'est effectué avec les référentiels et unités archivistiques de rattachement présents dans la solution logicielle Vitam. De fait, il est fortement recommandé de :
  - veiller à inclure des références (ex. service producteur, contrat d'entrée, etc.) existant dans la solution, de manière à éviter de possibles échecs lors des étapes de contrôles des données référentielles du processus d'entrée (opération « INGEST ») ;
  - ne pas envoyer dans le module de collecte des éléments sans valeurs, en particulier ceux qui sont obligatoires et doivent nécessairement être renseignés.

Cette action provoque la mise à jour du projet.
Lors de cette action, l'opération peut aboutir aux résultats suivants :

Statut	Motifs
Succès	Action réalisée sans rencontrer de problèmes particuliers.
Échec	Présence d'un champ inconnu dans la requête. Le projet n'existe pas.

3.4.2.2. Modification d'une transaction

La solution logicielle Vitam permet de modifier une transaction, et plus précisément de :

- modifier ses métadonnées,
- lui ajouter des métadonnées,
- supprimer des métadonnées.

Exemple : requête en vue de modifier des métadonnées d'une transaction dont l'identifiant est « aeeaaaaaachj3m7nabjocamcdqr2rqaaaaaq »

```
@tenant = 1

PUT {{url}}/collect-external/v1/transactions
accept: application/json
content-type: application/json
Content-Length: 401
X-Tenant-Id: {{tenant}}

{
  "#id": "aeeaaaaaachj3m7nabjocamcdqr2rqaaaaaq",
  "ArchivalAgencyIdentifier": "Vitam",
  "TransferringAgencyIdentifier": "RATP",
```

```
  "OriginatingAgencyIdentifier": "RATP",
  "SubmissionAgencyIdentifier": "RATP",
  "MessageIdentifier": "20200131-000001",
  "ArchivalAgreement":"IC-000001",
  "Comment": "RH - bulletins de salaire (février 2020)",
  "AcquisitionInformation":"Versement",
  "LegalStatus":"Public Archive"
}
```

Points d'attention :

- Le service de mise à jour d'une transaction fonctionne en mode « annule et remplace ».
- La plupart des champs d'une transaction étant facultatifs, il est recommandé d'indiquer les éléments obligatoires à la génération d'un SIP :
  - le contrat d'entrée (champ « ArchivalAgreement »),
  - l'identifiant du message (champ « MessageIdentifier),
  - le service producteur (champ « OriginatingAgencyIdentifier »),
  - le service d'archivage (champ « ArchivalAgencyIdentifier »),
  - le service responsable du transfert (« TransferringAgencyIdentifier »)
- Au terme de la version 6.RC :
  - ce service ne vérifie pas le caractère obligatoire d'un champ. Il est fortement recommandé de veiller à ne pas supprimer un champ obligatoire, de manière à éviter de possibles échecs lors des étapes de contrôles des données référentielles du processus d'entrée (opération « INGEST ») ;
  - on ne peut pas ajouter de règles de gestion dans une transaction ;
  - aucun contrôle n'est effectué avec les référentiels et unités archivistiques de rattachement présents dans la solution logicielle Vitam. De fait, il est fortement recommandé de :
    - veiller à inclure des références (ex. service producteur, contrat d'entrée, etc.) existant dans la solution, de manière à éviter de possibles échecs lors des étapes de contrôles des données référentielles du processus d'entrée (opération « INGEST ») ;
    - ne pas envoyer dans le module de collecte des éléments sans valeurs, en particulier ceux qui sont obligatoires et doivent nécessairement être renseignés.

Cette action provoque la mise à jour du projet.
Lors de cette action, l'opération peut aboutir aux résultats suivants :

Statut	Motifs
Succès	Action réalisée sans rencontrer de problèmes particuliers.
Échec	Présence d'un champ inconnu dans la requête. La transaction n'existe pas.

3.4.2.3. Modification des unités archivistiques

La solution logicielle Vitam permet de modifier des unités archivistiques par import d'un fichier au format .csv, et plus précisément de :

- modifier ses métadonnées,
- lui ajouter des métadonnées.

Points d'attention :

- En prérequis à la mise à jour des unités archivistiques, il faut avoir au préalable créé :
  - une transaction et le signaler dans l'API ;
  - les unités archivistiques qui sont référencées dans le fichier .csv ;
- Il n'est pas possible de :
  - supprimer une métadonnée en l'état actuel des développements ;
  - ajouter une unité archivistique lors de cette action de mise à jour.

Exemple : requête en vue de modifier des métadonnées d'unités archivistiques associées à une transaction dont l'identifiant est « aeeaaaaaachj3m7nabjocamcdqr2rqaaaaaq »

@transaction-id = aeeaaaaaachj3m7nabjocamcdqr2rqaaaaaq

PUT {{url}}/collect-external/v1/transactions/{{**transaction-id**}}/units
Accept: application/json
Content-Type: application/octet-stream
X-Tenant-Id: {{tenant}}
X-Access-Contract-Id: {{access-contract}}

< /path_tozip/stream.csv

Pour une transaction donnée peut être envoyé un fichier .csv contenant des métadonnées détaillant unitairement tout ou partie des unités archivistiques préalablement envoyées.

Le fichier .csv est composé de x colonnes[17] :

- File : chemin relatif à partir de l'emplacement où est positionnée l'unité archivistique faisant l'objet de la modification. Il s'agit d'une concaténation des intitulés des différentes unités archivistiques (colonne obligatoire) ;
- toute colonne correspondant à un champ du standard SEDA et nécessitant une modification et/ou un ajout de métadonnées (colonnes facultatives).

Points d'attention :

- le fichier .csv n'est pas obligatoirement intitulé « metadata.csv »
- l'ordre des premières colonnes ne doit pas être modifié ;
- une première ligne d'en-tête donnant le nom des colonnes doit être présente, chaque ligne décrivant ensuite une unité archivistique ;

17 Son modèle est identique au modèle de fichier .csv importé dans ReSIP, hors colonnes ID et ParentID qui ne sont pas supportées par le module de collecte en l'état actuel des développements.

- le séparateur entre les colonnes est le point-virgule, le séparateur de texte les guillemets doubles et l'encodage est « UTF-8 » ;
- le fichier .csv ne référence que des métadonnées propres aux unités archivistiques (métadonnées descriptives et de gestion).
- seul un objet peut être associé à un enregistrement. Ce format d'import ne permet pas *de facto* de gérer l'import de groupe d'objets techniques disposant de plusieurs objets aux usages différents devant être référencé par la même unité archivistique.
- concernant le nommage des colonnes :
  - pour les colonnes correspondant à des champs du standard SEDA, l'intitulé de la colonne doit correspondre à celui du champ dans le standard SEDA, précédé de « Management. » s'il s'agit d'une métadonnée de gestion (ex. « Management.AccessRule.Rule » pour une règle de communicabilité) ou de « Content » s'il s'agit d'une métadonnée descriptive (ex. « Content.DocumentType »). Toutefois, si le fichier d'import ne décrit que des métadonnées descriptives, la présence du préfixe « Content » est facultative ;
  - quand le schéma XML du standard SEDA propose une structure complexe de balises (par exemple pour décrire un auteur via l'objet XML <Writer> qui contient plusieurs balises XML comme FullName ou BirthName), il convient d'intituler la colonne de la manière suivante : Content.Writer.FullName ou Content.Writer.BirthName ;
  - quand un champ ou un objet XML est multivalué dans le standard SEDA (et qu'il est possible d'en décrire plusieurs dans le bordereau comme c'est le cas pour l'objet Writer par exemple), il convient de numéroter la colonne de la manière suivante : Content.Writer.0.FullName, Content.Writer.1.FullName ;
- concernant le contenu des colonnes :
  - la colonne File :
    - indique la position de l'unité archivistique dans l'arborescence de la transaction, depuis l'unité archivistique racine jusqu'à l'unité archivistique décrite. Il s'agit là d'une concaténation des intitulés des différentes unités archivistiques.
    - ne doit pas comprendre d'espace avant ou après les « \ » ;
    - doit correspondre à un chemin tel que décrit par l'explorateur de fichiers (avec des « \ » et non des « / ») ;
  - la colonne DescriptionLevel, si elle est présente, ne doit comprendre que les valeurs autorisées par le standard SEDA : Collection, Fonds, Series, SubSeries, RecordGrp, File, Item ;
  - les colonnes correspondant à des champs Date dans le standard SEDA doivent être formatées conformément à la norme ISO 8601 (AAAA-MM-JJ) ;
  - les références à des règles de gestion doivent se conformer aux identifiants de règles présents dans le SAE ;
  - les dates de début (Content.StartDate) doivent être antérieures aux dates de fin (Content.EndDate).

Exemple : fichier .csv de mise à jour des métadonnées (ajout ou modification des dates de début et de fin pour les

unités archivistiques intitulées « AU1 » et « AU2 »)

File;Content.DescriptionLevel;Content.Title;Content.StartDate;Content.EndDate
"content/AU1";"Item";"AU1";"1970-06-03";"1980-06-03"
"content/AU2";"Item";"AU2";"1970-06-03";"1980-06-03"
"content/AU3";"Item";"AU3";"";""
"content/AU4";"Item";"AU4";"";""

Cette action provoque la mise à jour des unités archivistiques dans la base de données MongoDB, dans la collection « Unit » (base *MetadataCollect*[18]).

Lors de cette action, l'opération peut aboutir aux résultats suivants :

Statut	Motifs
Succès	Action réalisée sans rencontrer de problèmes particuliers.
Avertissement	Le formatage du fichier .csv contient au moins une erreur (ex. date mal formatée, valeur attendue erronée, date de début postérieure à la date de fin, etc.)
Échec	Le fichier .csv n'est pas au format .csv. Le fichier .csv contient des erreurs dans la colonne File : • deux enregistrements contiennent le même chemin ; • le chemin est erroné ou introuvable. Action non réalisée pour cause de nom erroné ou de chemin introuvable dans la requête. La transaction n'existe pas ou est erronée. La transaction a un statut « READY », « SENDING », « SEND », « ACK_OK », « ACK_WARNING », « ACK_KO », « KO », « ABORTED ».

Elle n'est pas journalisée dans le journal des opérations.

Points d'attention : Au terme de la V.6 RC, le module de collecte :

- ne bloque pas l'opération de mise à jour de métadonnées si le fichier CSV comporte les erreurs suivantes :
  - il inverse les colonnes Title et DescriptionLevel ;
  - il ne contient pas au moins une colonne de métadonnées obligatoires (Title, DescriptionLevel) ;
  - il dispose d'un champ dont le contenu est mal formaté (ex. ReceivedDate écrite en chaîne de caractères) ;
  - il contient des virgules, des espaces, des pipes comme séparateurs de champs ;
  - le fichier contient des simples guillemets comme séparateurs de texte ;

18 Pour plus d'informations sur l'enregistrement de l'unité archivistique dans la base de données, il est recommandé de consulter le document *Modèle de données*.

- il contient un champ obligatoire non renseigné (Title) ;

Si le fichier CSV est erroné, une erreur peut être renvoyée par l'API, l'(les) unité(s) archivistique(s) en erreur sera(seront) ignorée(s) et seule(s) l'(les) unité(s) archivistique(s) sans erreur fera(feront) l'objet de la mise à jour ;

- ne permet pas de supprimer une métadonnée.

3.4.2.4. Suppression

La solution logicielle Vitam permet de :

- supprimer un projet de versement,
- supprimer une transaction.

Points d'attention : En prérequis à cette action, il faut avoir au préalable créé :

- un projet de versement et le signaler dans l'API, pour une suppression de projet ;
- une transaction et le signaler dans l'API, pour une suppression de transaction.

Exemple : requête en vue de supprimer un projet de versement dont l'identifiant est « aeeaaaaaachj3m7nabjocamcdqr2rqaaaaaq »

@project-id= *aeeaaaaaachj3m7nabjocamcdqr2rqaaaaaq*

```
DELETE {{url}}/collect-external/v1/projects/{{project-id}}
accept: application/json
content-type: application/json
X-Tenant-Id: {{tenant}}
X-Access-Contract-Id: {{access-contract}}
```

Exemple : requête en vue de supprimer une transaction dont l'identifiant est « aeeaaaaaachj3m7nabjocamcdqr2rqaaaaaq »

@transaction-id= *aeeaaaaaachj3m7nabjocamcdqr2rqaaaaaq*

```
DELETE {{url}}/collect-external/v1/transactions/{{transaction-id}}
accept: application/json
content-type: application/json
X-Tenant-Id: {{tenant}}
X-Access-Contract-Id: {{access-contract}}
```

Ces actions provoquent :

- pour la première, la suppression de l'enregistrement dans la base de données MongoDB, dans la collection « Project » (base *Collect*).
  Si le projet de versement est associé à une transaction qui, elle-même contient des archives (unités archivistiques, objets techniques), l'ensemble des archives sont supprimées de la base de données et des offres de stockage :

  - les unités archivistiques sont supprimées de la collection « Unit » (base *MetadataCollect*) ;
  - les groupes d'objets techniques sont supprimées de la collection « ObjectGroup » (base *MetadataCollect*).
  - les objets sont supprimés des offres de stockage ;

- pour la seconde, la suppression de l'enregistrement dans la base de données MongoDB, dans la collection « Transaction » (base *Collect*).
  Si la transaction contient des archives (unités archivistiques, objets techniques), l'ensemble des données associées à cette transaction sont supprimées de la base de données et des offres de stockage :
  - les unités archivistiques sont supprimées de la collection « Unit » (base *MetadataCollect*) ;
  - les groupes d'objets techniques sont supprimées de la collection « ObjectGroup » (base *MetadataCollect*).
  - les objets sont supprimés des offres de stockage.

  Mais le projet de versement ne sera pas supprimé.

Lors de cette action, l'opération peut aboutir aux résultats suivants :

Statut	Motifs
Succès	Action réalisée sans rencontrer de problèmes particuliers.
Échec	Le projet ou la transaction n'existe pas. Le projet ou la transaction n'est pas reconnu dans la requête.

3.4.2.5. Abandon

La solution logicielle Vitam permet également d'abandonner une transaction.

Point d'attention : En prérequis à cette action, il faut avoir au préalable créé une transaction et le signaler dans l'API.

Exemple : requête en vue d'abandonner une transaction dont l'identifiant est « aeeaaaaaachj3m7nabjocamcdqr2rqaaaaaq »

@transaction-id= *aeeaaaaaachj3m7nabjocamcdqr2rqaaaaaq*

```
PUT {{url}}/collect-external/v1/transactions/{{transaction-id}}/abort
Accept: application/json
Content-Type: application/json
X-Tenant-Id: {{tenant}}

{}
```

Cette action provoque :

- la modification de l'enregistrement dans la base de données MongoDB, dans la collection « Transaction » (base *Collect*) : la valeur du champ « Status » devient « ABORTED » ;
- la purge des données associées à la transaction, enregistrées dans les collections « Unit » et « ObjectGroup » (base *MetadataCollect*) et sur les offres de stockage, passé un certain délai configurable par tenant (60 minutes par défaut)[19].

Points d'attention :

- Une transaction peut être abandonnée uniquement lorsque son statut est égal à « OPEN », « READY », « ACK_KO », « KO ».
  Si elle a un statut égal à « SENDING », « SENT », « ACK_OK », « ACK_WARNING », elle ne peut pas l'être.
- La différence avec l'action de suppression est que l'abandon :
  - ne purge pas les archives directement, mais de manière automatisée, passé un certain délai ;
  - donne lieu à une mise à jour du statut de la transaction.

Lors de cette action, l'opération peut aboutir aux résultats suivants :

Statut	Motifs
Succès	Action réalisée sans rencontrer de problèmes particuliers.
Échec	La transaction n'existe pas. La transaction n'est pas reconnue dans la requête. La transaction a un statut égal à « SENDING », « SENT », « ACK_OK », « ACK_WARNING ».

3.4.2.6. Réouverture

La solution logicielle Vitam permet également de rouvrir (ou rééditer) une transaction.

Point d'attention : En prérequis à cette action, il faut avoir au préalable créé une transaction et le signaler dans l'API.

Exemple : requête en vue de réouvrir une transaction dont l'identifiant est « aeeaaaaaachj3m7nabjocamcdqr2rqaaaaaq »

@transaction-id= *aeeaaaaaachj3m7nabjocamcdqr2rqaaaaaq*

PUT {{url}}/collect-external/v1/transactions/aeeaaaaabohhxsvcaaxnqamfooffxkqaaaaq/reopen
Accept: application/json

19 Le paramétrage de la purge automatique s'effectue depuis le fichier collect.conf. Pour modifier le délai par défaut, il faut modifier les valeurs des paramètres « purgeTransactionThreadFrequency » et « purgeTransactionDelayInMinutes » pour le(s) tenant(s) concerné(s).

```
Content-Type: application/json
X-Tenant-Id: {{tenant}}
X-Access-Contract-Id: {{access-contract}}

{}
```

Cette action provoque la modification de l'enregistrement dans la base de données MongoDB, dans la collection « Transaction » (base *Collect*) : la valeur du champ « Status » devient « OPEN ».

Dès lors, il est à nouveau possible de :

- associer à la transaction rouverte des unités archivistiques, ainsi que des groupes d'objets techniques et des objets numériques ;
- modifier des unités archivistiques.

Point d'attention : Une transaction peut être rouverte uniquement lorsque son statut est égal à « READY », « ACK_KO », « KO ».
Si elle a un statut égal à « OPEN », « SENDING », « SENT », « ACK_OK », « ACK_WARNING », « ABORTED », elle ne peut pas l'être.

Lors de cette action, l'opération peut aboutir aux résultats suivants :

Statut	Motifs
Succès	Action réalisée sans rencontrer de problèmes particuliers.
Échec	La transaction n'existe pas. La transaction n'est pas reconnue dans la requête. La transaction a un statut égal à « OPEN », « SENDING », « SENT », « ACK_OK », « ACK_WARNING », « ABORTED ».

3.4.3. Utilisation dans VitamUI

L'APP « Collecte et préparation des versements » du front-office VitamUI fournie avec la solution logicielle Vitam permet :

- au moment de la création d'un projet de versement, d'abandonner ce dernier à tout moment du processus de création, au moyen d'un bouton « Annuler » ;
- dès qu'une transaction est créée, de l'abandonner et/ou de la rouvrir depuis la liste des transactions (ou versements) associées à un projet de versement donné.
  Dans le cas d'un abandon, les archives associées à la transaction sont supprimées, passé un certain délai configurable par tenant (60 minutes par défaut)[20]. Le projet de versement et la transaction restent visibles depuis les interfaces ;

20 Le paramétrage de la purge automatique s'effectue depuis le fichier collect.conf. Pour modifier le délai par défaut, il faut modifier les valeurs des paramètres « purgeTransactionThreadFrequency » et « purgeTransactionDelayInMinutes » pour le(s) tenant(s) concerné(s).

- de modifier des unités archivistiques par import d'un fichier .csv, depuis le détail d'un projet de versement.

Points d'attention : Au terme de la version 6 RC, il n'est pas possible, depuis les interfaces, de :

- modifier un projet préalablement créé ou une transaction ;
- supprimer un projet de versement ou une transaction. En cas d'abandon d'une transaction, le projet et la transaction restent visibles.

3.5. Transfert

3.5.1. Définitions

La solution logicielle Vitam permet de :

- clôturer une transaction (ou versement),
- la transférer depuis le module de collecte vers le back-office sous la forme d'un SIP conforme au SEDA 2.2.,
- si le transfert est en succès ou en avertissement, supprimer automatiquement les archives qui sont associées à la transaction.

L'opération de clôture, effectuée dans le module de collecte, n'est pas journalisée dans le journal des opérations, tandis que l'opération de transfert des archives sous la forme d'un SIP, envoyée par le module de collecte, est tracée dans le journal des opérations (opération de type « INGEST »).

3.5.2. Utilisation des API

Une fois le versement des données dans le module de collecte terminé, il est possible de :

- clôturer la transaction,
- générer un SIP pour transfert dans la solution logicielle Vitam.

Point d'attention : l'ordre d'utilisation des API doit être respecté en vue de transférer avec succès l'ensemble des métadonnées et objets constituant un versement vers la solution logicielle Vitam pour conservation.

3.5.2.1. Clôture d'une transaction

La solution logicielle permet de clôturer une transaction, une fois l'ensemble des archives liées à cette transaction envoyées dans le module de collecte.

Points d'attention :

- Cette action est un prérequis à l'envoi d'un SIP depuis le module de collecte vers la solution logicielle Vitam.
- Il faut avoir au préalable créé une transaction et la signaler dans l'API.

Exemple : requête de clôture

```
@transaction-id = aeeaaaaaaghiyso4ablmyal74slqwtqaaaaq
@unit-id = aeeaaaaaaghiyso4ablmyal74sndwiqaaaaq
@tenant = 1

POST  {{url}}/collect-external/v1/transactions/{{transaction-id}}/close
Accept: application/json
Content-Type: application/json
X-Tenant-Id: {{tenant}}
X-Access-Contract-Id: ContratTNR

{}
```

Cette action provoque la modification de l'enregistrement dans la base de données MongoDB, dans la collection « Transaction » (base *Collect*) : la valeur du champ « Status » est désormais « READY ».

Exemple : enregistrement de la transaction dans la collection « Transaction »


```
{
   "_id": "aeeaaaaaaghdvam4abgoyambfxnhd6aaaaaq",
   "Status": "READY",
   "Context": {
      "ArchivalAgreement": "IC-000001",
      "MessageIdentifier": "20220302-000005",
      "ArchivalAgencyIdentifier": "Vitam",
      "TransferringAgencyIdentifier": "AN",
      "OriginatingAgencyIdentifier": "MICHEL_MERCIER",
      "SubmissionAgencyIdentifier": "MICHEL_MERCIER",
      "Comment": "Versement du service producteur : Cabinet de Michel Mercier"
   },
    "CreationDate": "2022-10-15T15:50:06.153",
    "LastUpdate": "2022-11-15T15:50:06.153",
    "ProjectId": "aeaaaaaaaaha7iacabzrsamepp45blyaaaaq",
   "_tenant": 1
}
```


Dès lors, il n'est plus possible de :

- associer à cette transaction des unités archivistiques, ainsi que des groupes d'objets techniques et des objets numériques ;
- modifier des unités archivistiques.

Lors de cette action, l'opération peut aboutir aux résultats suivants :

Statut	Motifs

Succès	Action réalisée sans rencontrer de problèmes particuliers.
Échec	- La transaction associée n'existe pas ; - La transaction associée a déjà été clôturée.

Point d'attention : S'il s'avère nécessaire de modifier le contenu de la transaction, il est toujours possible à ce stade de la rouvrir[21].

3.5.2.2. Envoi du SIP

Pour une transaction donnée, une fois celle-ci clôturée, le module de collecte permet de générer un SIP et de le transférer au moyen d'une opération de type « INGEST ».

Point d'attention :

- En prérequis à l'envoi du SIP, il faut avoir au préalable clôturé la transaction (son statut doit être égal à « READY ») et signaler cette dernière dans l'API.

Exemple : requête d'envoi du SIP vers la solution logicielle Vitam pour conservation

```
@transaction-id = aeeaaaaaaghiyso4ablmyal74slqwtqaaaaq
@unit-id = aeeaaaaaaghiyso4ablmyal74sndwiqaaaaq
@tenant = 1

POST  {{url}}/collect-external/v1/transactions/{{transaction-id}}/send
Accept: application/json
Content-Type: application/json
X-Tenant-Id: {{tenant}}
X-Access-Contract-Id: ContratTNR

{

}
```

Cette action provoque :

- si elle est en erreur technique :
  - la modification de l'enregistrement dans la base de données MongoDB, dans la collection « Transaction » (base *Collect*) : la valeur du champ « Status » est « KO » ;
- si elle est conforme :
  - l'initialisation d'une opération de type « INGEST »,

21 Se référer à la sous-section « Réouverture » du présent document.

- la modification de l'enregistrement dans la base de données MongoDB, dans la collection « Transaction » (base *Collect*) :
  - la valeur du champ « Status » est :
    - « SENDING » tant que l'opération de type « INGEST » n'a pas été initialisée par la solution logicielle Vitam,
    - « SENT » quand l'opération de type « INGEST » est en cours ;
  - un champ correspondant à l'identifiant de l'opération d'entrée est ajouté (VitamOperationId).Une fois l'opération de type « INGEST » terminée, le statut est :
    - si l'opération est en succès, la valeur du champ « Status » est désormais « ACK_OK » ; si l'opération est en avertissement, la valeur du champ « Status » est « ACK_WARNING » ;
    - si l'opération est en erreur, la valeur du champ « Status » est « ACK_KO ».

Lors de cette action, l'opération peut aboutir aux résultats suivants :

Statut	Motifs
Succès	Action réalisée sans rencontrer de problèmes particuliers.
Échec	- La transaction associée n'existe pas ; - La transaction associée n'a pas été clôturée ; - La transaction ne contient pas d'archives associées (unités archivistiques et groupes d'objets techniques).

Point d'attention : Au terme de la V6.RC, les archives transférées avec succès ou en avertissement (statut égal à « ACK_OK » et « ACK_WARNING ») pour archivage sont purgées du module de collecte passé un certain délai configurable par tenant (60 minutes par défaut)[22].

3.5.3. Utilisation dans VitamUI

L'APP « Collecte et préparation des versements » du front-office VitamUI fournie avec la solution logicielle Vitam utilise les API de clôture et de génération d'un SIP pour envoi dans la solution logicielle Vitam.
Ces services sont disponibles :

- depuis la page permettant de visualiser l'ensemble des archives d'un projet de versement, où il est possible de :
  - « Valider » un versement, action correspondant à la clôture du versement,
  - « Verser » un versement, action correspondant à l'envoi du SIP.

22 Le paramétrage de la purge automatique s'effectue depuis le fichier collect.conf. Pour modifier le délai par défaut, il faut modifier les valeurs des paramètres « purgeTransactionThreadFrequency » et « purgeTransactionDelayInMinutes » pour le(s) tenant(s) concerné(s).

- depuis la page permettant de visualiser l'ensemble des transactions (ou versements) associées à un projet de versement, où il est possible de :
  - « Accéder aux archives »,
  - « Valider » un versement, action correspondant à la clôture du versement,
  - « Verser » un versement, action correspondant à l'envoi du SIP,
  - « Editer » un versement, action correspondant à la réouverture du versement en vue, notamment, de le modifier,
  - « Abandonner » un versement, action correspondant à l'abandon du versement,

Certaines de ces actions entraînent la purge des archives passé un certain délai. Il s'agit des actions suivantes :

- « Abandonner »
- « Verser », si le résultat de cette action est un versement en succès ou en avertissement.

Actions disponibles	**Statut correspondant**		**Purge automatique ?**
	Dans le front-office	**Dans le back-office**	
Valider	Validé	READY	
Verser	Préparation et envoi du SIP	SENDING	
	Envoyé, en cours de traitement du SAE	SENT	
	Versé en succès	ACK_OK	OUI
	Versé	ACK_WARNING	OUI
	Échec du versement	ACK_KO	
	Erreur technique	KO	
Éditer	Ouvert en édition	OPEN	
Abandonner	Abandonné	ABORTED	OUI

4. Conseils de mise en œuvre

À l'issue de cette phase de réalisation de fonctionnalités concernant le module de collecte, l'équipe du programme Vitam est en mesure de fournir plusieurs recommandations de mise en œuvre.

4.1. Comment formaliser les données dans les API du module de collecte ?

Il est possible de verser séparément dans le module de collecte selon un formalisme spécifique au format JSON :

- les métadonnées correspondant à l'en-tête du message ArchiveTransfer,
- les métadonnées décrivant les unités archivistiques,
- les métadonnées techniques.

Chaque appel API[23] permettant de verser ces métadonnées inclut une liste d'éléments, clés ou vocabulaires que l'on souhaite intégrer dans le module de collecte.

Si ces éléments doivent respecter le formalisme attendu par le SEDA et l'ontologie de la solution logicielle Vitam (s'il s'agit d'une chaîne de caractères, d'une date, d'un élément répétable, etc.), il n'est pas nécessaire de les ordonnancer comme le demande le SEDA.

Si certains éléments, issus du SEDA et/ou du système externe, ne doivent pas être envoyés, il n'est pas nécessaire de les référencer dans les différents appels API[24].

Certains contrôles, disponibles au moment du transfert dans la solution logicielle Vitam n'ont pas été implémentés. Il est fortement recommandé de ne pas envoyer dans le module de collecte des éléments sans valeurs, en particulier ceux qui sont obligatoires et doivent nécessairement être renseignés.

Généralités

Dans un enregistrement JSON, un élément (ou vocabulaire) est désigné par son **nom**.

Exemple : l'élément suivant se nomme Description.

```
"Description": "Ceci est une description"
```

Points d'attention :

- Si un élément (ou vocabulaire), présent dans un enregistrement, n'existe pas dans l'ontologie, Il est fortement recommandé de l'avoir préalablement créé dans le référentiel[25].

23 Ces appels API sont décrits dans le chapitre 3 du présent document, dans la sous-section 3.1.2 « Utilisation des API ».

24 La partie ci-dessous explicite la grammaire attendue par le format JSON dans le cadre d'un appel API. Des précisions sur le formalisme attendu est également disponible dans le document *Modèle de données*.

25 Pour plus d'informations sur les vocabulaires, consulter le document *Ontologie*.

- Les vocabulaires issus du SEDA sont nommés de la même manière qu'ils le sont dans le standard (ex. l'élément « Tag » défini dans le SEDA est référencé sous ce nom dans l'ontologie et doit être indiqué comme tel).
- Il existe néanmoins quelques exceptions à cette règle :
  - le bloc Management doit être déclaré sous le nom #management. La modélisation des catégories de règle de gestion attend :
    - un sous-bloc « Rules » pouvant contenir 0 à n règles de gestion, non défini par le SEDA,
    - un sous-bloc « Inheritance » pouvant bloquer des règles de gestion, non défini par le SEDA. Ce sous-bloc peut également contenir l'élément « PreventRulesId » correspondant au champ « RefNonRuleId » du SEDA.
  - si le bloc Event porte le nom « Event », ses sous-blocs doivent être renommés pour se conformer à l'ontologie :
    - « EventIdentifier » en « evId »,
    - « EventTypeCode » en « evTypeProc »,
    - « EventType » en « evType »,
    - « EventDateTime » en « evDateTime »,
    - « EventDetail » en « evTypeDetail »,
    - « Outcome » en « outcome »,
    - « OutcomeDetail » en « outDetail »,
    - « OutcomeDetailMessage » en « outMessg »,
    - « EventDetailData » en « evDetData ».
  - les éléments Title et Description, s'ils contiennent des attributs, seront intitulés « Title_ » et « Description_ » et devront définir des propriétés[26].

Types

L'élément (ou vocabulaire) est associé à un **type** particulier[27]. On distingue plusieurs types JSON possibles :

- « string » : texte ;
- « number » : nombre, entier ou décimal ;
- « integer » : nombre entier ;
- « boolean » : booléen dont la valeur est true ou false ;
- « object » : objet ;
- « array » : liste ou tableau de valeurs textuelles.

Il doit alors être cohérent avec celui du vocabulaire tel qu'il est déclaré dans l'ontologie[28] :

26 Pour plus d'informations sur ces modélisations particulières, il est recommandé de consulter le document Modèle de données, chapitre 5.1, « Collection Unit ».

27 Les types des éléments propres au SEDA sont listés dans l'annexe 2 « Types JSON ». Il est conseillé de se reporter à cette annexe, afin de typer correctement les éléments.

28 La présence de crochets dans le tableau de correspondances indique que le vocabulaire employé dans l'enregistrement peut ou doit se présenter sous la forme d'un tableau ou « array », pouvant contenir un type de valeur en particulier. Le vocabulaire représenté entre crochets est également répétable.

Type d'indexation dans l'ontologie	Type correspondant dans un profil d'unité archivistique		Commentaires
	Vocabulaire interne	Vocabulaire externe	
TEXT	string ou [string]	[string]	
KEYWORD	string ou [string]	[string]	
DATE	string ou [string] + pattern date	[string] + pattern date	
LONG	number ou integer [number] ou [integer]	[number] ou [integer]	
DOUBLE	number ou [number]	[number]	
BOOLEAN	boolean ou [boolean]	[boolean]	
GEO_POINT	string	[string]	L'équipe Vitam n'a pas investigué sur les usages de ces deux types d'indexation.
ENUM	[string] + pattern énumératif	[string] + pattern énumératif	

Par analogie au SEDA et au langage XML, il convient de prêter attention aux éléments suivants :

- sera qualifié en objet (« object ») un élément contenant des sous-éléments, par exemple : Management, Writer, Keyword ;
- sera qualifié en tableau (ou « array ») :
  - un élément répétable, tel que Tag ou OriginatingAgencyArchiveUnitIdentifier,
  - un vocabulaire externe ;
- certains éléments du SEDA (PreventInheritance, NeedAuthorization, NeedReassessingAuthorization) doivent contenir un booléen.

On trouvera essentiellement les types suivants :

- pour les **informations d'en-tête** : les éléments sont uniquement de type « string » ou « array » ;
- pour une **unité archivistique** :
  - pour les *vocabulaires internes*, propres au SEDA, les principaux types rencontrés sont : « string », « array » et « object », auxquels s'ajoute un unique « boolean » et un unique « integer »[29].
  - quant aux *vocabulaires externes*, ajoutés pour répondre à des besoins et transferts spécifiques, il faut les identifier systématiquement comme des « array » (ou tableaux),

29 Les types des éléments propres au SEDA sont listés dans l'annexe 2 « Types JSON ». du présent document.

c’est-à-dire des éléments répétables. Ces tableaux peuvent inclure ensuite un type particulier de chaînes : texte, entier, décimal ou booléen.

- Pour un **groupe d’objets techniques** :
  - pour les *vocabulaires internes*, propres au SEDA, les principaux types rencontrés sont : « string », « array » et « object », auxquels s’ajoutent quelques « integer » ou « number ».

Les éléments de type « array » et « object » ont une structuration plus complexe qu’un type simple :

- un type simple se définit par un nom, auquel est associé une seule valeur. Ce type est associé à un élément dont la cardinalité est 0-1 ou 1-1.

Exemple : les vocabulaires Description, GpsAltitude et NeedAuthorization sont des éléments simples. Attendant un entier, Size est caractérisé par un item de type « integer », tandis que NeedAuthorization est de type « boolean ».

"Description": "Ceci est une description"
"GpsAltitude": 390
"NeedAuthorization": true

- un élément de type « array » peut contenir une liste ou un tableau de valeurs ou d’objets. Ce type est associé à un élément dont la cardinalité est 0-n ou 1-n.

Exemple : les vocabulaires externes NomDuCapitaine, AgeDuCapitaine et le vocabulaire interne Tag sont dotés d’un type « array » dans l’enregistrement d’une unité archivistique. Attendant un entier, AgeDuCapitaine est caractérisé par un item de type « integer ».

"NomDuCapitaine" : ["Capitaine Fracasse"],
"AgeDuCapitaine" : [1],
"Tag" : ["Marine", "Capitaine Fracasse"]

- un élément de type « object » définit des sous-éléments. Ce type peut être associé à un tableau de sous-éléments (ex. « RegisteredAgent »).

Exemple :

- *l’élément RegisteredAgent est un tableau d’objets qui contient les sous-propriétés ou sous-éléments FirstName, FullName, BirthName.*

"RegisteredAgent": [{
"FirstName": "description",
"FullName": "description",
"BirthName": "description"
}]

- *l’élément BirthPlace contient les sous-propriétés ou sous-éléments Geogname, Address, PostalCode, City, Region, Country.*

```
"BirthPlace": {
            "Geogname": "Toronto [Ontario]",
            "Address": "123 my Street",
            "PostalCode": "CD255",
            "City": "Toronto",
            "Region": "Ontario",
            "Country": "Canada"
        }
```

Enfin, en fonction du type d'indexation, les valeurs doivent être enregistrées

- soit entre guillemets (string)
- soit sans guillemets (integer, number, boolean).

Exemple : les vocabulaires Title, Size et NeedReassessingAuthorization sont des éléments simples : le premier est un string, le second un integer et le dernier un boolean.

```
"Title": "Ceci est une description"
"Size": 390
"NeedReassessingAuthorization": false
```

Cas particulier

Les éléments **« Title » et « Description »**, s'ils définissent un attribut ou sont répétables, doivent prendre la forme d'objet.

Exemple : sont autorisés les vocabulaires Title et Description avec attributs, avec une cardinalité 1 – n.

```
"Title_": {
    "fr": "Ceci est un titre en français",
    "en": "This is a title in English"
 }
"Description_": {
    "fr": "Ceci est une description en français",
    "en": "This is a content in English"
 }
```

Le bloc définissant les **règles de gestion** contient également des particularités. Il doit être déclaré sous le nom #management. La modélisation des catégories de règle de gestion attend :

- un sous-bloc « Rules » pouvant contenir 0 à n règles de gestion,
- un sous-bloc « Inheritance » pouvant bloquer des règles de gestion[30].

Exemple : Le bloc AppraisalRule déclare deux règles différentes dans le sous-bloc Rules, « APP-00001 » et « APP-00002 », et définit un blocage sur la catégorie de règle avec le sous-bloc PreventInheritance.

```
"#management": {
```

30 Pour plus d'informations sur ces modélisations particulières, il est recommandé de consulter le document *Modèle de données*, chapitre 5.1, « Collection Unit ».

```
    "AppraisalRule": {
      "Rules": [ {
         "Rule": "APP-00001",
         "StartDate": "2015-01-01",
         "EndDate": "2095-01-01"
         },
         {
         "Rule": "APP-00002"
         } ],
       "Inheritance": {
         "PreventInheritance": true,
         "PreventRulesId": []
         },
      "FinalAction": "Keep"
}
```

4.2. Comment sont gérés les statuts d'une transaction ?

Le module de collecte attribue un certain nombre de statuts à une transaction. Ces statuts sont générés à la suite d'une action :

Actions disponibles	**Statut correspondant**		**Modification possible ?**	**Abandon possible ?**	**Purge automatique ?**
	Dans le front-office	**Dans le back-office**			
Créer la transaction	Ouvert en édition	OPEN	OUI	OUI	
Clôturer / Valider	Validé	READY		OUI	
Verser / Envoyer	Préparation et envoi du SIP	SENDING			
	Envoyé, en cours de traitement du SAE	SENT			
	Versé en succès	ACK_OK			OUI
	Versé	ACK_WARNING			OUI
	Échec du versement	ACK_KO		OUI	
	Erreur technique	KO		OUI	
Éditer / Rouvrir	Ouvert en édition	OPEN	OUI		
Abandonner	Abandonné	ABORTED			OUI

Annexe 1 : Exemples de données entrantes

Nota bene : les cas présentés ci-dessous sont des exemples fictifs.

Métadonnées d'en-tête

```
{
 "ArchivalAgencyIdentifier": "Vitam",
 "TransferringAgencyIdentifier": "AN",
 "OriginatingAgencyIdentifier": "MICHEL_MERCIER",
 "SubmissionAgencyIdentifier": "MICHEL_MERCIER",
 "MessageIdentifier": "20220302-000005",
 "ArchivalAgreement":"IC-000001",
 "Comment": "Versement du service producteur : Cabinet de Michel Mercier"
}
```

Unités archivistiques

```
{
   "DescriptionLevel": "RecordGrp",
   "Title": "Discours de Michel Mercier",
   "#management": {
    "AccessRule": {
        "Rules": [
           {
              "Rule": "ACC-00001",
              "StartDate": "2000-01-01"
           }
        ]
     }
    }
}
```

Objets

```
{
 "FileInfo": {
   "Filename": "MonFichier.txt"
 }
}
```

Annexe 2 : Types JSON

Pour les éléments propres au SEDA, le tableau suivant précise les types de certains d'entre eux :

- Métadonnées d'en-tête :

	obligatoire	**string**	**number**	**boolean**	**object**	**array**
ArchivalAgreement	x	x				
MessageIdentifier	x	x				
Comment		x[31]				
OriginatingAgencyIdentifier	x	x				
SubmissionAgencyIdentifier		x				
ArchivalAgencyIdentifier	x	x				
TransferringAgencyIdentifier	x	x				
ArchivalProfile		x				
AcquisitionInformation		x				

- Métadonnées descriptives et de gestion d'une unité archivistique :

	obligatoire	**string**	**number**	**boolean**	**object**	**array**
ArchiveUnitProfile		x				
#management[32]	x				x	
AccessRule					x	
AppraisalRule					x	
StorageRule					x	
ReuseRule					x	
ClassificationRule					x	
HoldRule					x	
Rules					x	x
Rule		x				
StartDate		x[33]				

31 En l'état actuel des développements, le champ Comment est non répétable, alors que, dans le SEDA, il l'est.

32 #management est à employer en lieu et place de la balise Management pour se conformer aux attendus de la solution logicielle Vitam.

33 Avec un pattern date.

EndDate		x[34]				
FinalAction[35]		x				
Inheritance					x	
PreventInheritance				x		
PreventRulesId						x[36]
HoldEndDate		x[37]				
HoldOwner		x				
HoldReason		x				
HoldReassessingDate		x[38]				
PreventRearrangement				x		
ClassificationOwner		x				
ClassificationReassessingDate		x[39]				
NeedReassessingAuthorization				x		
ClassificationLevel		x				
ClassificationAudience		x				
Logbook						x
NeedAuthorization				x		
DescriptionLevel	x	x				
Title	x	x				x
Title_					x	
FilePlanPosition		x				x
SystemId		x				x
OriginatingSystemId		x				x
ArchivalAgencyArchiveUnitIdentifier		x				x
OriginatingAgencyArchiveUnitIdentifier		x				x
TransferringAgencyArchiveUnitIdentifier		x				x
Description		x				x
Description_					x	

34 Avec un pattern date.
37 Avec un pattern date.
36 Tableau contenant des éléments de type string.
35 Simple énumération.
38 Avec un pattern date.
39 Avec un pattern date.

CustodialHistory					x	
CustodialHistoryItem		x				x
Type		x				
DocumentType		x				
Language		x[40]				x
DescriptionLanguage		x[41]				
Status		x				
Version		x				
Tag		x				x
Keyword					x	x
KeywordContent		x				
KeywordReference		x				
KeywordType		x				
Coverage					x	
Spatial		x				x
Temporal		x				x
Juridictional		x				x
OriginatingAgency					x	
SubmissionAgency					x	
Identifier		x				
AuthorizedAgent					x	x
Writer					x	x
Addressee					x	x
Recipient					x	x
Transmitter					x	x
Sender					x	x
FirstName		x				
BirthName		x				
FullName		x				
GivenName		x				
Gender		x				
BirthDate		x[42]				

40 Le SEDA attend plus précisément un pattern langue.
41 Le SEDA attend plus précisément un pattern langue.

DeathDate		x[43]				
BirthPlace					x	
DeathPlace					x	
Geogname		x				
Address		x				
PostalCode		x				
City		x				
Region		x				
Country		x				
Nationality		x				
Corpname		x				
Identifier		x				
Function		x				
Activity		x				
Position		x				
Role		x				
Mandate		x				
RelatedObjectReference					x	
IsVersionOf					x	
Replaces					x	
Requires					x	
IsPartOf					x	
References					x	
ArchiveUnitRefId		x				
DataObjectReference					x	
DataObjectReferenceId		x				
DataObjectGroupReferenceId		x				
RepositoryArchiveUnitPID		x				
RepositoryObjectPID		x				
CreatedDate		x				
TransactedDate		x				
AcquiredDate		x				

42 Avec un pattern date.

43 Avec un pattern date.

SentDate		x				
ReceivedDate		x				
RegisteredDate		x				
StartDate		x				
EndDate		x				
Event					x	
evId[44]		x				
evTypeProc[45]		x				
evType[46]		x				
EvDateTime[47]		x				
evTypeDetail[48]		x				
outcome[49]		x				
outDetail[50]		x				
outMessg[51]		x				
evDetData[52]		x				
Signature					x	
Signer					x	
Validator					x	
ValidationTime		x				
MasterData		x				
ReferencedObject					x	
SignedObjectId		x				
SignedObjectDigest		x				
Gps					x	

44 evId est à employer en lieu et place de la balise EventIdentifier dans un schéma de contrôle pour se conformer aux attendus de la solution logicielle Vitam.

45 evTypeProc est à employer en lieu et place de la balise EventTypeCode dans un schéma de contrôle pour se conformer aux attendus de la solution logicielle Vitam.

46 evType est à employer en lieu et place de la balise EventType dans un schéma de contrôle pour se conformer aux attendus de la solution logicielle Vitam.

47 evDateTime est à employer en lieu et place de la balise EventDateTime dans un schéma de contrôle pour se conformer aux attendus de la solution logicielle Vitam.

48 evTypeDetail est à employer en lieu et place de la balise EventDetail dans un schéma de contrôle pour se conformer aux attendus de la solution logicielle Vitam.

49 outcome est à employer en lieu et place de la balise Outcome dans un schéma de contrôle pour se conformer aux attendus de la solution logicielle Vitam.

50 outDetail est à employer en lieu et place de la balise OutcomeDetail dans un schéma de contrôle pour se conformer aux attendus de la solution logicielle Vitam.

51 outMessg est à employer en lieu et place de la balise OutcomeDetailMessage dans un schéma de contrôle pour se conformer aux attendus de la solution logicielle Vitam.

52 evDetData est à employer en lieu et place de la balise EventDetailData dans un schéma de contrôle pour se conformer aux attendus de la solution logicielle Vitam.

GpsVersionId		x				
GpsAltitude			x			
GpsAltitudeRef		x				
GpsLatitude		x				
GpsLongitude		x				
GpsLongitudeRef		x				
GpsDateStamp		x				

Annexe 3 : Liste des points d'API

	Description	**Permission**	**Verbe**	**EndPoint**

project	Création d'un projet de versement	project:create	POST	/collect-external/v1/projects/
	Récupère la liste des projets de versement	project:read	GET	/collect-external/v1/projects/
	Récupère un projet de versement	project:id:read	GET	/collect-external/v1/projects/{projectId}/
	Récupère la liste des projets de versement par critère de recherche	project:query:read	GET	project:query:read
	Mise à jour d'un projet de versement	project:update	PUT	/collect-external/v1/projects
	Supprime un projet de versement	project:id:delete	DELETE	/collect-external/v1/projects/{projectId}/
	Récupérer la liste des transactions du projet	project:id:transactions	GET	/collect-external/v1/projects/{projectId}/ transactions/
	Récupère toutes les unités archivistiques associées à un projet	project:id:units	GET	/collect-external/v1/projects/{projectId}/ units/
transaction	Création de la transaction	transaction:create	POST	/collect-external/v1/projects/{projectId}/ transactions/
	Mise à jour d'une transaction	transaction:update	PUT	/collect-external/v1/transactions/
	Récupère une transaction	transaction:id:read	GET	/collect-external/v1/transactions/ {transactionId}/
	Clôture de la transaction	transaction:close	POST	/collect-external/v1/transactions/ {transactionId}/close/
	Envoi de la transaction	transaction:send	POST	/collect-external/v1/transactions/ {transactionId}/send/
	Abandonner une transaction	transaction:abort	PUT	/collect-external/v1/transactions/ {transactionId}/abort/
	Rouvrir une transaction	transaction:reopen	PUT	/collect-external/v1/transactions/ {transactionId}/reopen/
	Charge les binaires en lot	transaction:zip:create	POST	/collect-external/v1/transactions/ {transactionId}/upload/
	Crée une unité archivistique	transaction:unit:create	POST	/collect-external/v1/transactions/ {transactionId}/units/
	Récupère toutes les unités	transaction:unit:read	GET	/collect-external/v1/transactions/

	archivistiques			{transactionId}/units/
	Supprime une transaction	transaction:id:delete	DELETE	/collect-external/v1/transactions/{transactionId}/
units	Crée ou modifie un groupe d'objets techniques	transaction:object:upsert	POST	/collect-external/v1/units/{unitId}/objects/{usage}/{version}/
	Insère ou modifie un objet binaire	transaction:binary:upsert	POST	/collect-external/v1/units/{unitId}/objects/{usage}/{version}/binary/
	Récupère une unité archivistique	transaction:unit:id:read	GET	/collect-external/v1/units/{{unit-id}}/
	Mettre à jour les unités archivistiques	transaction:id:units:update	PUT	/collect-external/v1/transactions/{transactionId}/units/
	Télécharge un usage/version du binaire d'un groupe d'objets techniques	transaction:binary:read	GET	/collect-external/v1/units/{unitId}/objects/{usage}/{version}/binary/
objects	Récupère un groupe d'objets techniques	transaction:object:read	GET	/collect-external/v1/objects/{gotId}/