VITAM - Modèle de workflow

Version 0.15.1

VITAM

avr. 26, 2017

Table des matières

CHAPITRE 1

Sommaire

1.1 Workflow d'ingest/d'entrée

1.1.1 Introduction

Ce document décrit le processus (workflow) d'entrée mis en place dans la solution logicielle Vitam, l'implémentation mise en place pour celui-ci dans la version bêta et la V1 de cette solution.

1.1.2 Processus d'entrée (vision métier)

Un workflow est un processus composé d'étapes (macro-workflow), elles-mêmes composées d'une liste d'actions à exécuter de manière séquentielle, une seule fois ou répétées sur une liste d'éléments (micro-workflow).

Chaque étape, chaque action peuvent avoir les statuts suivants :

- OK : le traitement associé s'est passé correctement. Le workflow continue.
- Warning : le traitement associé a généré un avertissement (e.g. le format de l'objet est mal déclaré dans le bordereau de versement). Le workflow continue.
- KO : le traitement associé a généré une erreur métier. Le workflow s'arrête si le modèle d'execution est bloquant (cf. ci-dessous).
- FATAL : le traitement associé a généré une erreur technique. Le workflow s'arrête.

Chaque action peut avoir les modèles d'éxécutions suivants (toutes les étapes sont par défaut bloquantes) :

- Bloquant
  - Si une action est identifiée en erreur, l'étape en cours est alors arrêtée et le workflow passe à la derniere étape de finalisation de l'entrée. Un accusé de reception est généré et le statut du processus d'entrée passe à « erreur ».
- Non bloquant
  - Si une action est identifiée en erreur, le reste des actions est exécuté et le statut de l'étape correspondant passe à « erreur ». L'étape en cours est alors arrêtée et le workflow passe à la derniere étape de finalisation de l'entrée. Un accusé de reception est généré et le statut du processus d'entrée passe à « erreur ».

Le processus d'entrée débute lors du lancement du chargement d'un Submission Information Package dans la solution Vitam. De plus, toutes les étapes et actions sont journalisées dans le journal des opérations. Les étapes et actions associées ci-dessous décrivent le processus d'entrée (clé et description de la clé associée dans le journal des opérations) tel qu'implémenté dans la version bêta de la solution logicielle :

1.1.2.1 Contrôles préalables à l'entrée (STP_SANITY_CHECK_SIP)

- Contrôle sanitaire (SANITY_CHECK_SIP)
  - **Règle** : vérification de l'absence de virus dans le SIP.
  - **Type** : bloquant.
  - **Statuts** :
    - OK : aucun virus n'est détecté dans le SIP (SANITY_CHECK_SIP.OK=Succès du contrôle sanitaire : aucun virus détecté)
    - KO : un ou plusieurs virus ont été detectés dans le SIP (SANITY_CHECK_SIP.KO=Échec du contrôle sanitaire du SIP : fichier détecté comme infecté)
    - FATAL : la vérification de la présence de virus dans le SIP n'a pas pu être faite suite à une erreur système (SANITY_CHECK_SIP.FATAL=Erreur fatale lors du contrôle sanitaire du SIP)
- Contrôle du format du conteneur du SIP (CHECK_CONTAINER)
  - **Règle** : Vitam vérifie le format du SIP via un outil d'identification de format
  - **Formats acceptés** : .zip, .tar, .tar.gz, .tar.bz2
  - **Type** : bloquant.
  - **Statuts** :
    - OK : le conteneur du SIP est au bon format (CHECK_CONTAINER.OK=Succès du contrôle de format du conteneur du SIP)
    - KO : le conteneur du SIP n'est pas au bon format (CHECK_CONTAINER.KO=Échec du contrôle de format du conteneur du SIP)
    - FATAL : la vérification du format du conteneur du SIP n'a pas pu être faite suite à une erreur système liée à l'outil d'identification des formats (CHECK_CONTAINER.FATAL=Erreur fatale lors du processus du contrôle de format du conteneur du SIP)

1.1.2.2 Réception dans vitam (STP_UPLOAD_SIP) : Etape de réception du SIP dans Vitam

- **Type** : bloquant.
- **Statuts** :
  - OK : le SIP a été reçu dans Vitam(STP_UPLOAD_SIP.OK=Succès du processus de téléchargement du SIP)
  - KO : le SIP n'a pas été reçu dans Vitam (STP_UPLOAD_SIP.KO=Échec du processus de téléchargement du SIP)
  - FATAL : la réception du SIP dans Vitam n'a pas été possible suite à une erreur système, e.g. serveur indisponible (STP_UPLOAD_SIP.FATAL=Erreur Fatale lors du processus de téléchargement du SIP)

1.1.2.3 Contrôle du SIP (STP_INGEST_CONTROL_SIP)

- Vérification globale du SIP (CHECK_SEDA) : Vérification de la cohérence physique du SIP
  - **Type de manifeste accepté** : le manifeste est obligatoire dans le SIP, doit être nommé manifest.xml et doit être conforme au schéma xsd par défaut fourni avec le standard SEDA v. 2.0.
  - **Type** : bloquant.
  - **Statuts** :
    - OK : le SIP est présent, nommé manifest.xml et conforme au schéma xsd par défaut fourni avec le standard SEDA v.2.0. (CHECK_SEDA.OK=Succès de la vérification globale du SIP)

        - KO : le manifeste est introuvable dans le SIP ou n'a pas d'extension .xml (CHECK_SEDA.NO_FILE.KO=Échec de la vérification globale du SIP : le manifeste est introuvable dans le SIP)
        - KO : le manifeste n'est pas au format XML (CHECK_SEDA.NOT_XML_FILE.KO=Échec de la vérification globale du SIP : le manifeste de versement au mauvais format)
        - KO : le manifeste ne respecte pas le schéma par défaut fourni avec le standard SEDA v.2.0 (CHECK_SEDA.NOT_XSD_VALID.KO=Échec de la vérification globale du SIP : manifeste non conforme au schéma SEDA 2.0)
        - FATAL : le manifeste n'a pas pu être contrôlé suite à une erreur système (CHECK_SEDA.FATAL=Erreur fatale lors de la vérification globale du SIP)
- Vérification des usages des groupes d'objets (CHECK_MANIFEST_DATAOBJECT_VERSION)
    - **Règle** : tous les objets décrits dans le manifeste du SIP doivent déclarer un usage conforme à la liste des usages acceptés dans la solution logicielle
    - **Types d'usages acceptés** : original papier (PhysicalMaster), original numérique (BinaryMaster), diffusion (Dissemination), vignette (Thumbnail), contenu brut (TextContent)
    - **Type** : bloquant.
    - **Statuts** :
        - OK : les objets contenus dans le SIP déclarent tous dans le manifeste un usage cohérent avec ceux acceptés (CHECK_MANIFEST_DATAOBJECT_VERSION.OK=Succès de la vérification des usages des groupes d'objets)
        - KO : un ou plusieurs objets contenus dans le SIP déclarent dans le manifeste un usage incohérent avec ceux acceptés (CHECK_MANIFEST_DATAOBJECT_VERSION.KO=Échec de la vérification des usages des groupes d'objets)
        - FATAL : les usages déclarés dans le manifeste pour les objets contenus dans le SIP n'ont pas pu être contrôlés suite à une erreur système (CHECK_MANIFEST_DATAOBJECT_VERSION.FATAL=Erreur fatale lors de la vérification des usages des groupes d'objets)
- Vérification du nombre d'objets (CHECK_MANIFEST_OBJECTNUMBER)
    - **Règle** : le nombre d'objets reçus dans la solution Vitam doit être strictement égal au nombre d'objets déclaré dans le manifeste du SIP
    - **Type** : bloquant.
    - **Statuts** :
        - OK : le nombre d'objets reçus dans la solution logicielle est strictement égal au nombre d'objets déclaré dans le manifeste du SIP (CHECK_MANIFEST_OBJECTNUMBER.OK=Succès de la vérification du nombre d'objets)
        - KO : le nombre d'objets reçus dans la solution logicielle est inférieur ou supérieur au nombre d'objets déclaré dans le manifeste du SIP (CHECK_MANIFEST_OBJECTNUMBER.KO=Échec de la vérification du nombre d'objets)
- Vérification de la cohérence du bordereau (CHECK_MANIFEST)
    - **Règle** : cette action permet la création des journaux de cycle de vie des unités archivistiques (ArchiveUnit) et des groupes d'objets (ObjectGroup), la vérification de la présence de cycles dans les arboresences des ArchiveUnits, la création de l'arbre d'ordre d'indexation et l'extraction des métadonnées contenues dans la balise ManagementMetadata du manifeste pour le calcul des règles de gestion.

- **Type** : bloquant.
- **Statuts** :
  - OK : les journaux de cycles de vie des ArchiveUnits et des ObjectGroups ont été créés avec succès, aucune récursivité n'a été detectée dans l'arborescence des ArchiveUnits (CHECK_MANIFEST.OK=Contrôle du bordereau réalisé avec succès)
  - KO : Une récursivité a été détectée dans l'arborescence des ArchiveUnits (CHECK_MANIFEST.KO=Échec de contrôle du bordereau)
  - FATAL : la vérification de la cohérence du bordereau n'a pas pu être réalisée suite à une erreur système, e.g. les journaux de cycle de vie n'ont pas pu être créés (CHECK_MANIFEST.FATAL=Erreur fatale lors de contrôle du bordereau)

- Vérification de la cohérence entre objets, groupes d'objets et unités archivistiques (CHECK_CONSISTENCY)
  - **Règle** : Chaque objet ou groupe d'objets doit être référencé par un ArchiveUnit, les objets sans groupe d'objets mais référencés par un ArchiveUnit sont rattachés chacun à un groupe d'objets.
  - **Type** : bloquant.
  - **Statuts** :
    - OK : Aucun objet ou groupe d'objet n'est orphelin (i.e. non référencé par une ArchiveUnit) et tous les objets sont rattachés à un groupe d'objets (CHECK_CONSISTENCY.OK=Succès de la vérification de la cohérence entre objets, groupes d'objets et unités archivistiques)
    - KO : Au moins un objet ou groupe d'objet est orphelin (i.e. non référencé par une ArchiveUnit) (CHECK_CONSISTENCY.KO=Échec de la vérification de la cohérence entre objets, groupes d'objets et unités archivistiques)
    - FATAL : la vérification de la cohérence entre objets, groupes d'objets et unités archivistiques n'a pas pu être réalisée suite à une erreur système (CHECK_CONSISTENCY.FATAL=Erreur fatale lors de la vérification de la cohérence entre objets, groupes d'objets et unités archivistiques)

1.1.2.4 Contrôle et traitements des objets (STP_OG_CHECK_AND_TRANSFORME)

- Vérification de l'intégrité des objets (CHECK_DIGEST)
  - **Règle** : vérification de la cohérence entre l'empreinte de l'objet calculée par la solution logicielle Vitam et celle déclarée dans le manifeste. Si l'empreinte déclarée dans le manifeste n'a pas été calculée avec l'algorithme SHA-512, alors le système recalcule une empreinte avec cette algorithme. C'est celle-ci qui sera enregistrée dans le système.
  - **Algorithmes autorisés en entrée** : MD5, SHA-1, SHA-256, SHA-512
  - **Type** : bloquant.
  - **Statuts** :
    - OK : tous les objets reçus sont identiques aux objets attendus. Tous les objets disposent désormais d'une empreinte calculée avec l'algorithme SHA-512 (CHECK_DIGEST.OK=Succès de la vérification de lintégrité des objets)
    - KO : au moins un objet reçu n'est pas identique aux objets attendus (CHECK_DIGEST.KO=Échec de la vérification de lintégrité des objets)
    - FATAL : la vérification de l'intégrité des objets n'a pas pu être réalisée suite à une erreur système, e.g. algorithme inconnu (CHECK_DIGEST.FATAL=Erreur fatale lors de la vérification des objets)
    - WARNING : tous les objets reçus sont identiques aux objets attendus, mais au moins un objet a une empreinte déclarée dans le manifeste non calculée par l'algorithme SHA-512 (CHECK_DIGEST.WARNING=Avertissement lors de la vérification de lempreinte)

- Identification des formats (OG_OBJECTS_FORMAT_CHECK)
  - **Règle** : Vitam identifie les formats de chaque objet présent dans le SIP, afin de garantir une information homogène et objective. Cette action met en œuvre un outil d'identification prenant l'objet en entrée et fournissant des informations de format en sortie. Ces informations sont comparées les formats identifiés dans le référentiel des formats interne au système et avec celles déclarées dans le manifeste. En cas d'incohérence entre la déclaration de l'application versante et le format identifié par le système, le SIP sera tout de même accepté, générant un warning. Vitam se servira alors des informations qu'il a lui même identifiées et non celles de l'application versante.
  - **Type** : bloquant.
  - **Statuts** :
    - OK : l'identification s'est bien passée, les formats identifiés sont référencés dans le référentiel interne et les informations sont cohérentes avec celles déclarées dans le manifeste (OG_OBJECTS_FORMAT_CHECK.OK=Succès de la vérification des formats)
    - KO : le format identifié n'est pas référencé dans le référentiel interne, ou aucun format n'a été trouvé pour un objet (OG_OBJECTS_FORMAT_CHECK.KO=Échec de la vérification des formats)
    - FATAL : l'identification des formats n'a pas été réalisée suite à une erreur système (OG_OBJECTS_FORMAT_CHECK.FATAL=Erreur fatale lors de la vérification des formats)
      - WARNING : l'identification s'est bien passée, les formats identifiés sont référencés dans le référentiel interne mais les informations ne sont pas cohérentes avec celles déclarées dans le manifeste (OG_OBJECTS_FORMAT_CHECK.WARNING=Avertissement lors de la vérification des formats)

1.1.2.5 Contrôle et traitements des unités archivistiques (STP_OG_CHECK_AND_TRANSFORME)

- Application des règles de gestion et calcul des dates d'échéances (UNITS_RULES_COMPUTE)
  - **Règle** : calcul des dates d'échéance des ArchiveUnits à la racine des DescriptiveMetadata du manifeste si elles existent dans le manifeste (Si elles n'existent pas, elles sont récupérées depuis la balise ManagementMetadata du manifeste). Calcul des dates d'échéance des ArchiveUnits filles si elles existent dans le manifeste. Le référentiel utilisé pour ce calcul est le référentiel des règles de gestion.
  - **Type** : bloquant.
  - **Statuts** :
    - OK : les règles de gestion sont référencées dans le référentiel interne et ont été appliquées avec succès (UNITS_RULES_COMPUTE.OK=Succès du calcul des dates d'échéance)
    - KO : au moins une règle de gestion déclarée dans le manifeste n'est pas référencée dans le référentiel interne (UNITS_RULES_COMPUTE.KO=Échec du calcul des dates d'échéance)

1.1.2.6 Préparation de la prise en charge (STP_STORAGE_AVAILABILITY_CHECK)

- Vérification de la disponibilité de l'offre de stockage (STORAGE_AVAILABILITY_CHECK)
  - **Type** : bloquant.
  - **Statuts** :
    - OK : l'offre de stockage est accessible et dispose d'assez d'espace pour stocker le contenu du SIP (STORAGE_AVAILABILITY_CHECK.OK=Succès de la vérification de la disponibilité de l'offre de stockage)
    - KO : l'offre de stockage n'est pas disponible ou ne dispose pas d'assez d'espace pour stocker le contenu du SIP (STORAGE_AVAILABILITY_CHECK.KO=Échec de la vérification de la disponibilité de l'offre de stockage)
    - FATAL : la vérification de la disponibilité de l'offre de stockage n'a pas pu être réalisée suite à une erreur système (STORAGE_AVAILABILITY_CHECK.FATAL=Erreur fatale lors de la vérification de la disponibilité de l'offre de stockage)

1.1.2.7 Rangement des objets (STP_OG_STORING)

- Enregistrement des objets sur l'offre de stockage (OG_STORAGE)
  - **Type** : Bloquant.
  - **Statuts** :
    - OK : tous les objets contenus dans le SIP ont été stockés dans l'offre de stockage (OG_STORAGE.OK=Succès du rangement des objets et groupes d'objets)
    - KO : au moins un des objets contenus dans le SIP n'a pas pu être stocké dans l'offre de stockage (OG_STORAGE.KO=Échec du rangement des objets et groupes d'objets)
    - FATAL : l'enregistrement des objets sur l'offre de stockage n'a pas pu être réalisé suite à une erreur système (OG_STORAGE.FATAL=Erreur fatale lors du rangement des objets et groupes d'objets)
- Indexation des métadonnées des groupes d'objets (OG_METADATA_INDEXATION)
  - **Règle** : les métadonnées liées aux groupes d'objets sont indexées, e.g. la taille des objets, l'empreinte des objets, les métadonnées liées aux formats (Type MIME, PUID, etc.)
  - **Type** : bloquant.
  - **Statuts** :
    - OK : les métadonnées des groupes d'objets ont été indexées avec succès (OG_METADATA_INDEXATION.OK=Succès de l'indexation des métadonnées des objets et groupes d'objets)
    - KO : les métadonnées des groupes d'objets n'ont pas pu être indexées (OG_METADATA_INDEXATION.KO=Échec de l'indexation des métadonnées des objets et groupes d'objets)
    - FATAL : l'indexation des métadonnées des groupes d'objets n'a pas pu être réalisée suite à une erreur système (OG_METADATA_INDEXATION.FATAL=Erreur fatale lors de l'indexation des métadonnées des objets et groupes d'objets)
- Sécurisation des métadonnées des groupes d'objets (OG_METADATA_STORAGE)
  - **Règle** : les métadonnées liées aux groupes d'objets sont stockées dans l'offre de stockage afin de les sécuriser
  - **Type** : bloquant.
  - **Statuts** :
    - OK : les métadonnées des groupes d'objets ont été sécurisées avec succès (OG_METADATA_STORAGE.OK=Succès de l'enregistrement des métadonnées des groupes d'‘objets)
    - KO : les métadonnées des groupes d'objets n'ont pas pu être sécurisées (OG_METADATA_STORAGE.KO=Échec de l'enregistrement des métadonnées des objets et groupes d'objets)
- Sécurisation du journal des cycles de vie des groupes d'objets (COMMIT_LIFE_CYCLE_OBJECT_GROUP) (post Bêta)
  - **Règle** : Suite à l'indexation des métadonnées liées aux groupe d'objets, les journaux de cycle de vie des groupes d'objets sont sécurisés en base (Avant cette étape, les journaux de cycle de vie des groupes d'objets sont dans une collection temporaire afin de garder une cohérence entre les métadonnées indexées et les JCV lors d'une entrée en succès ou en échec)
  - **Type** : bloquant.
  - **Statuts** :
    - OK : La sécurisation s'est correctement déroulée (COMMIT_LIFE_CYCLE_OBJECT_GROUP.OK=Succès de la sécurisation du journal du cycle de vie des groupes d'objets)
    - FATAL : La sécurisation du journal du cycle de vie n'a pas pu être réalisée suite à une erreur système (COMMIT_LIFE_CYCLE_OBJECT_GROUP.FATAL=Erreur fatale lors de la sécurisation du journal du cycle de vie des groupes d'objets)

1.1.2.8 Rangement des unites archivistiques (STP_UNIT_STORING)

- Indexation des métadonnées des unités archivistiques (UNIT_METADATA_INDEXATION)
  - **Type** : bloquant.
  - **Statuts** :
    - OK : les métadonnées des unités archivistiques ont été indexées avec succès (UNIT_METADATA_INDEXATION.OK=Succès de l'indexation des métadonnées des unités archivistiques)
    - KO : les métadonnées des unités archivistiques n'ont pas pu être indexées (UNIT_METADATA_INDEXATION.KO=Échec de l'indexation des métadonnées des unités archivistiques)
    - FATAL : l'indexation des métadonnées des unités archivistiques n'a pas pu être réalisée suite à une erreur système (UNIT_METADATA_INDEXATION.FATAL=Erreur fatale lors de l'indexation des métadonnées des unités archivistiques)
- Sécurisation des métadonnées des unités archivistiques (UNIT_METADATA_STORAGE)
  - **Type** : bloquant.
  - **Statuts** :
    - OK : les métadonnées des unités archivistiques ont été stockées avec succès (UNIT_METADATA_STORAGE.OK=Succès de l'enregistrement des métadonnées des unités archivistiques)
    - KO : les métadonnées des unités archivistiques n'ont pas pu être stockées (UNIT_METADATA_STORAGE.KO=Échec de l'enregistrement des métadonnées des unités archivistiques)
- Sécurisation du journal des cycles de vie des unités archivistiques (COMMIT_LIFE_CYCLE_UNIT) (post Bêta)
  - **Règle** : Suite à l'indexation des métadonnées liées aux unités archivistiques, les journaux de cycle de vie des unités archivistiques sont sécurisés en base (Avant cette étape, les journaux de cycle de vie des unités archivistiques sont dans une collection temporaire afin de garder une cohérence entre les métadonnées indexées et les JCV lors d'une entrée en succès ou en échec)
  - **Type** : bloquant.
  - **Statuts** :
    - OK : La sécurisation s'est correctement déroulée (COMMIT_LIFE_CYCLE_UNIT.OK=Succès de la sécurisation du journal du cycle de vie des unités archivistiques)
    - FATAL : La sécurisation du journal du cycle de vie n'a pas pu être réalisée suite à une erreur système (COMMIT_LIFE_CYCLE_UNIT.FATAL=Erreur fatale lors de la sécurisation du journal du cycle de vie des unités archivistiques)

1.1.2.9 Registre des fonds (STP_ACCESSION_REGISTRATION)

- Alimentation du registre des fonds (ACCESSION_REGISTRATION)
  - **Règle** : le registre des fonds est alimenté par service producteur.
  - **Type** : bloquant.
  - **Statuts** :
    - OK : le registre des fonds est correctement alimenté (ACCESSION_REGISTRATION.OK=Succès de l'alimentation du registre des fonds)
    - KO : le registre des fonds n'a pas pu être alimenté (ACCESSION_REGISTRATION.KO=Échec de l'alimentation du registre des fonds)
    - FATAL : l'alimentation du registre des fonds n'a pas pu être réalisée suite à une erreur système (ACCESSION_REGISTRATION.FATAL=Erreur fatale lors de l'alimentation du registre des fonds)

1.1.2.10 Finalisation de l'entrée (STP_INGEST_FINALISATION)

- Notification de la fin de l'opération d'entrée (ATR_NOTIFICATION)
  - **Règle** : une fois toutes les étapes passées avec succès ou lorsqu'une étape est en échec, cette étape est lancée. Elle gènere un message de réponse (ArchiveTransferReply ou ATR), le stocke dans l'offre de stockage et l'envoie au service versant.
  - **Type** : non bloquant.
  - **Statuts** :
    - OK : Le message de réponse a été correctement généré, stocké dans l'offre de stockage et envoyé au service versant (ATR_NOTIFICATION.OK=Succès de la notification à l'opérateur de versement)
    - KO : Le message de réponse n'a pas été correctement généré, stocké dans l'offre de stockage ou reçu par le service versant (ATR_NOTIFICATION.KO=Échec de la notification à l'opérateur de versement)
    - FATAL : la notification de la fin de l'opération n'a pas pu être réalisée suite à une erreur système (ATR_NOTIFICATION.FATAL=Erreur fatale lors de la notification à l'opérateur de versement)
- Mise en cohérence des journaux de cycle de vie (ROLL_BACK) (post Bêta)
  - **Règle** : une fois toutes les étapes passées avec succès ou lorsqu'une étape est en échec, cette étape est lancée suite à la notification de la fin d'opération d'entrée afin de purger les collections temporaire des journaux de cycle de vie.
  - **Type** : bloquant.
  - **Statuts** :
    - OK : La purge s'est correctement déroulée (ROLL_BACK.OK=Succès de la mise en cohérence des journaux de cycle de vie)
    - FATAL : la purge n'a pas pu être réalisée suite à une erreur système (ROLL_BACK.FATAL=Erreur fatale lors la mise en cohérence des journaux de cycle de vie)

1.1.3 Structure du Workflow (Implémenté en V1)

Le workflow actuel mis en place dans la solution Vitam est défini dans l'unique fichier "DefaultIngestWorkflow.json". Il décrit le processus d'entrée (hors Ingest externe) pour entrer un SIP, indexer les métadonnées et stocker les objets contenues dans le SIP.

```
{
  "id": "DefaultIngestWorkflow",
  "comment": "Default Ingest Workflow V6",
  "steps": [
    {
      "workerGroupId": "DefaultWorker",
      "stepName": "STP_INGEST_CONTROL_SIP",
      "behavior": "BLOCKING",
      "distribution": {
        "kind": "REF",
        "element": "SIP/manifest.xml"
      },
      "actions": [
        {
          "action": {
            "actionKey": "CHECK_SEDA",
            "behavior": "BLOCKING"
          }
        },
```

```
        {
          "action": {
            "actionKey": "CHECK_MANIFEST_DATAOBJECT_VERSION",
            "behavior": "BLOCKING"
          }
        },
        {
          "action": {
            "actionKey": "CHECK_MANIFEST_OBJECTNUMBER",
            "behavior": "NOBLOCKING"
          }
        },
        {
          "action": {
            "actionKey": "CHECK_MANIFEST",
            "behavior": "BLOCKING",
            "out": [
              {
                "name": "unitsLevel.file",
                "uri": "WORKSPACE:UnitsLevel/ingestLevelStack.json"
              },
              {
                "name": "mapsBDOtoOG.file",
                "uri": "WORKSPACE:Maps/BDO_TO_OBJECT_GROUP_ID_MAP.json"
              },
              {
                "name": "mapsBDO.file",
                "uri": "WORKSPACE:Maps/BINARY_DATA_OBJECT_ID_TO_GUID_MAP.json"
              },
              {
                "name": "mapsObjectGroup.file",
                "uri": "WORKSPACE:Maps/OBJECT_GROUP_ID_TO_GUID_MAP.json"
              },
              {
                "name": "mapsObjectGroup.file",
                "uri": "WORKSPACE:Maps/OG_TO_ARCHIVE_ID_MAP.json"
              },
              {
                "name": "mapsBDOtoVersionBDO.file",
                "uri": "WORKSPACE:Maps/BDO_TO_VERSION_BDO_MAP.json"
              },
              {
                "name": "mapsUnits.file",
                "uri": "WORKSPACE:Maps/ARCHIVE_ID_TO_GUID_MAP.json"
              },
              {
                "name": "globalSEDAParameters.file",
                "uri": "WORKSPACE:ATR/globalSEDAParameters.json"
              }
            ]
          }
        },
        {
          "action": {
            "actionKey": "CHECK_CONSISTENCY",
            "behavior": "NOBLOCKING",
            "in": [
              {
```

```
                "name": "mapsBDOtoOG.file",
                "uri": "WORKSPACE:Maps/OG_TO_ARCHIVE_ID_MAP.json"
              },
              {
                "name": "mapsBDOtoOG.file",
                "uri": "WORKSPACE:Maps/OBJECT_GROUP_ID_TO_GUID_MAP.json"
              }
            ]
          }
        }
      ]
    },
    {
      "workerGroupId": "DefaultWorker",
      "stepName": "STP_OG_CHECK_AND_TRANSFORME",
      "behavior": "BLOCKING",
      "distribution": {
        "kind": "LIST",
        "element": "ObjectGroup"
      },
      "actions": [
         {
          "action": {
            "actionKey": "CHECK_DIGEST",
            "behavior": "BLOCKING",
            "in": [
              {
                "name": "algo",
                "uri": "VALUE:SHA-512"
              }
            ]
          }
        },
        {
          "action": {
            "actionKey": "OG_OBJECTS_FORMAT_CHECK",
            "behavior": "BLOCKING"
          }
        }
      ]
    },
    {
      "workerGroupId": "DefaultWorker",
      "stepName": "STP_UNIT_CHECK_AND_PROCESS",
      "behavior": "BLOCKING",
      "distribution": {
        "kind": "LIST",
        "element": "Units"
      },
      "actions": [
        {
          "action": {
            "actionKey": "UNITS_RULES_COMPUTE",
            "behavior": "BLOCKING"
          }
        }
      ]
    },
```

```
    {
      "workerGroupId": "DefaultWorker",
      "stepName": "STP_STORAGE_AVAILABILITY_CHECK",
      "behavior": "BLOCKING",
      "distribution": {
        "kind": "REF",
        "element": "SIP/manifest.xml"
      },
      "actions": [
        {
          "action": {
            "actionKey": "STORAGE_AVAILABILITY_CHECK",
            "behavior": "BLOCKING"
          }
        }
      ]
    },
    {
      "workerGroupId": "DefaultWorker",
      "stepName": "STP_OG_STORING",
      "behavior": "BLOCKING",
      "distribution": {
        "kind": "LIST",
        "element": "ObjectGroup"
      },
      "actions": [
        {
          "action": {
            "actionKey": "OG_STORAGE",
            "behavior": "BLOCKING"
          }
        },
        {
          "action": {
            "actionKey": "OG_METADATA_INDEXATION",
            "behavior": "BLOCKING"
          }
        }
      ]
    },
    {
      "workerGroupId": "DefaultWorker",
      "stepName": "STP_UNIT_STORING",
      "behavior": "BLOCKING",
      "distribution": {
        "kind": "LIST",
        "element": "Units"
      },
      "actions": [
        {
          "action": {
            "actionKey": "UNIT_METADATA_INDEXATION",
            "behavior": "BLOCKING"
          }
        }
      ]
    },
    {
```

```
      "workerGroupId": "DefaultWorker",
      "stepName": "STP_ACCESSION_REGISTRATION",
      "behavior": "BLOCKING",
      "distribution": {
        "kind": "REF",
        "element": "SIP/manifest.xml"
      },
      "actions": [
        {
          "action": {
            "actionKey": "ACCESSION_REGISTRATION",
            "behavior": "BLOCKING",
            "in": [
              {
                "name": "mapsUnits.file",
                "uri": "WORKSPACE:Maps/ARCHIVE_ID_TO_GUID_MAP.json"
              },
              {
                "name": "mapsBDO.file",
                "uri": "WORKSPACE:Maps/OBJECT_GROUP_ID_TO_GUID_MAP.json"
              },
              {
                "name": "mapsBDO.file",
                "uri": "WORKSPACE:Maps/BDO_TO_BDO_INFO_MAP.json"
              },
              {
                "name": "globalSEDAParameters.file",
                "uri": "WORKSPACE:ATR/globalSEDAParameters.json"
              }
            ]
          }
        }
      ]
    },
    {
      "workerGroupId": "DefaultWorker",
      "stepName": "STP_INGEST_FINALISATION",
      "behavior": "FINALLY",
      "distribution": {
        "kind": "REF",
        "element": "SIP/manifest.xml"
      },
      "actions": [
        {
          "action": {
            "actionKey": "ATR_NOTIFICATION",
            "behavior": "BLOCKING",
            "in": [
              {
                "name": "mapsUnits.file",
                "uri": "WORKSPACE:Maps/ARCHIVE_ID_TO_GUID_MAP.json",
                "optional": "true"
              },
              {
                "name": "mapsBDO.file",
                "uri": "WORKSPACE:Maps/BINARY_DATA_OBJECT_ID_TO_GUID_MAP.json",
                "optional": "true"
              },
```

```
                {
                  "name": "mapsBDOtoOG.file",
                  "uri": "WORKSPACE:Maps/BDO_TO_OBJECT_GROUP_ID_MAP.json",
                  "optional": "true"
                },
                {
                  "name": "mapsBDOtoVersionBDO.file",
                  "uri": "WORKSPACE:Maps/BDO_TO_VERSION_BDO_MAP.json",
                  "optional": "true"
                },
                {
                  "name": "globalSEDAParameters.file",
                  "uri": "WORKSPACE:ATR/globalSEDAParameters.json",
                  "optional": "true"
                }
              ],
              "out": [
                {
                  "name": "atr.file",
                  "uri": "WORKSPACE:ATR/responseReply.xml"
                }
              ]
            }
          }
        ]
      }
  ]
}
```

D'une façon synthétique, le workflow est décrit de cette façon :

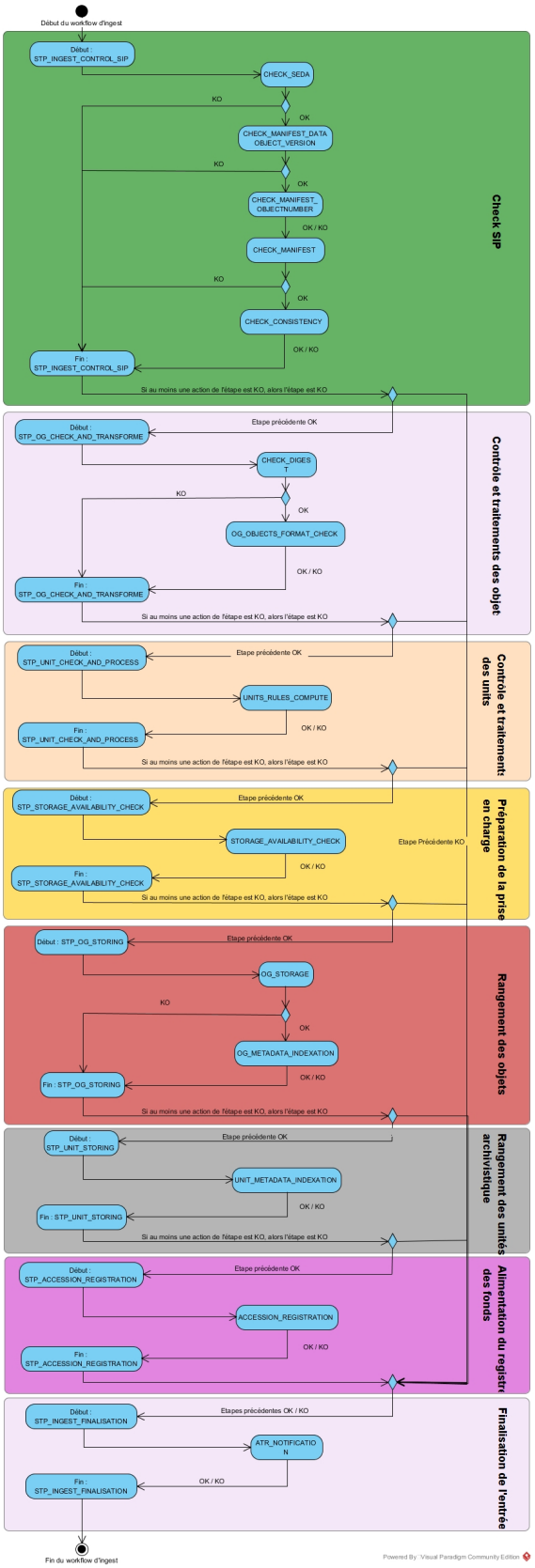

Fig. 1.1 – Diagramme d'état / transitions du workflow d'ingest

- **Step 1** - STP_INGEST_CONTROL_SIP : Check SIP / distribution sur REF GUID/SIP/manifest.xml
  - CHECK_SEDA (CheckSedaActionHandler.java) :
    - Test de l'existence du manifest.xml,
    - Test de l'existence d'un fichier unique à la racine du SIP
    - Test de l'existence d'un dossier unique à la racine, nommé "Content" (insensible à la casse)
    - Validation XSD du manifeste,
    - Validation de la structure du manifeste par rapport au schema par défaut fourni avec le standard SEDA v. 2.0.
  - CHECK_MANIFEST_DATAOBJECT_VERSION (CheckVersionActionHandler.java) :
    - Vérification des usages des objets.
  - CHECK_MANIFEST_OBJECTNUMBER (CheckObjectsNumberActionHandler.java) :
    - Comptage des objets (BinaryDataObject) dans le manifest.xml en s'assurant de l'absence de doublon, que le nombre d'objets reçus est strictement égal au nombre d'objets attendus
    - Création de la liste des objets dans le workspace GUID/SIP/content/,
    - Comparaison du nombre et des URI des objets contenus dans le SIP avec ceux définis dans le manifeste.
  - CHECK_MANIFEST (ExtractSedaActionHandler.java) :
    - Extraction des ArchiveUnits, des BinaryDataObject,
    - Création des journaux de cycle de vie des ArchiveUnits et des ObjectGroup,
    - Vérification de la présence de cycles dans les arboresences des Units,
    - Création de l'arbre d'ordre d'indexation,
    - Extraction des métadonnées contenues dans le bloc ManagementMetadata du manifeste pour le calcul des règles de gestion.
  - CHECK_CONSISTENCY (CheckObjectUnitConsistencyActionHandler.java) :
    - Extraction des BinaryDataObject du manifest.xml et création de la MAP (table de concordance) des Id BinaryDataObject / Génération GUID (de ces mêmes BinaryDataObject),
    - Extraction des ArchiveUnit du manifest.xml et création de la MAP des id ArchiveUnit / Génération GUID (de ces mêmes ArchiveUnit),
    - Contrôle des références dans les ArchiveUnit des Id BinaryDataObject,
    - Vérification de la cohérence objet/unit,
    - Stockage dans le Workspace des BinaryDataObject et des ArchiveUnit.
- **Step 2** - STP_OG_CHECK_AND_TRANSFORME : Contrôle et traitements des objets / distribution sur LIST GUID/BinaryDataObject
  - CHECK_DIGEST (CheckConformityActionPlugin.java) :
    - Contrôle de l'objet binaire correspondant : la taille et l'empreinte du BinaryDataObject.
  - Calcul d'une empreinte avec l'algorithme SHA-512 si l'empreinte du manifeste n'a pas été calculée avec cet algorithme
  - OG_OBJECTS_FORMAT_CHECK (FormatIdentificationActionPlugin.java) :
    - Identification du format des BinaryDataObject,
  - Vérification de l'existence du format identifié dans le référentiel des formats
    - Consolidation de l'information du format dans l'ObjectGroup correspondant si nécessaire.
- **Step 3** - STP_UNIT_CHECK_AND_PROCESS : Contrôle et traitements des units / distribution sur LIST GUID
  - UNITS_RULES_COMPUTE (UnitsRulesComputePlugin.java) :
    - vérification de l'existence de la règle dans le référentiel des règles de gestion

    - calcul des échéances associées à chaque ArchiveUnit.
- **Step 4** - STP_STORAGE_AVAILABILITY_CHECK : Préparation de la prise en charge / distribution REF GUID/SIP/manifest.xml
    - STORAGE_AVAILABILITY_CHECK (CheckStorageAvailabilityActionHandler.java) :
        - Calcul de la taille totale des objets à stocker,
        - Contrôle de la taille totale des objets à stocker par rapport à la capacité des offres de stockage pour une stratégie et un tenant donnés.
- **Step 5** - STP_OG_STORING : Rangement des objets
    - OG_STORAGE (StoreObjectGroupActionPlugin.java) :
        - Écriture des objets sur l'offre de stockage des BinaryDataObject des ObjectGroup.
    - OG_METADATA_INDEXATION (IndexObjectGroupActionPlugin.java) :
        - Enregistrement en base des métadonnées des ObjectGroup.
- **Step 6** - STP_UNIT_STORING : Rangement des unités archivistique / distribution sur LIST GUID/Units
    - UNIT_METADATA_INDEXATION (IndexUnitActionPlugin.java) :
        - Transformation sous la forme Json des ArchiveUnits et intégration du GUID Unit et du GUID ObjectGroup,
        - Enregistrement en base des métadonnées des ArchiveUnits.
- **Step 7** - STP_ACCESSION_REGISTRATION : Alimentation du registre des fonds
    - ACCESSION_REGISTRATION (AccessionRegisterActionHandler.java) :
        - Création/Mise à jour et enregistrement des collections AccessionRegisterDetail et AccessionRegisterSummary concernant les archives prises en compte, par service producteur.
- **Step 8 et finale** - STP_INGEST_FINALISATION : Finalisation de l'entrée. Cette étape est obligatoire et sera toujours exécutée, en dernière position.
    - ATR_NOTIFICATION (TransferNotificationActionHandler.java) :
        - Génération de l'ArchiveTransferReply.xml (peu importe le statut du processus d'entrée, l'ArchiveTransferReply est obligatoirement généré),
        - Stockage de l'ArchiveTransferReply dans les offres de stockage.

1.1.4 Structure du fichier Properties du Worflow

Le fichier Properties permet de définir la structure du Workflow pour les étapes et actions réalisées dans le module d'Ingest Interne, en excluant les étapes et actions réalisées dans le module d'Ingest externe.

La structure du fichier est la suivante :

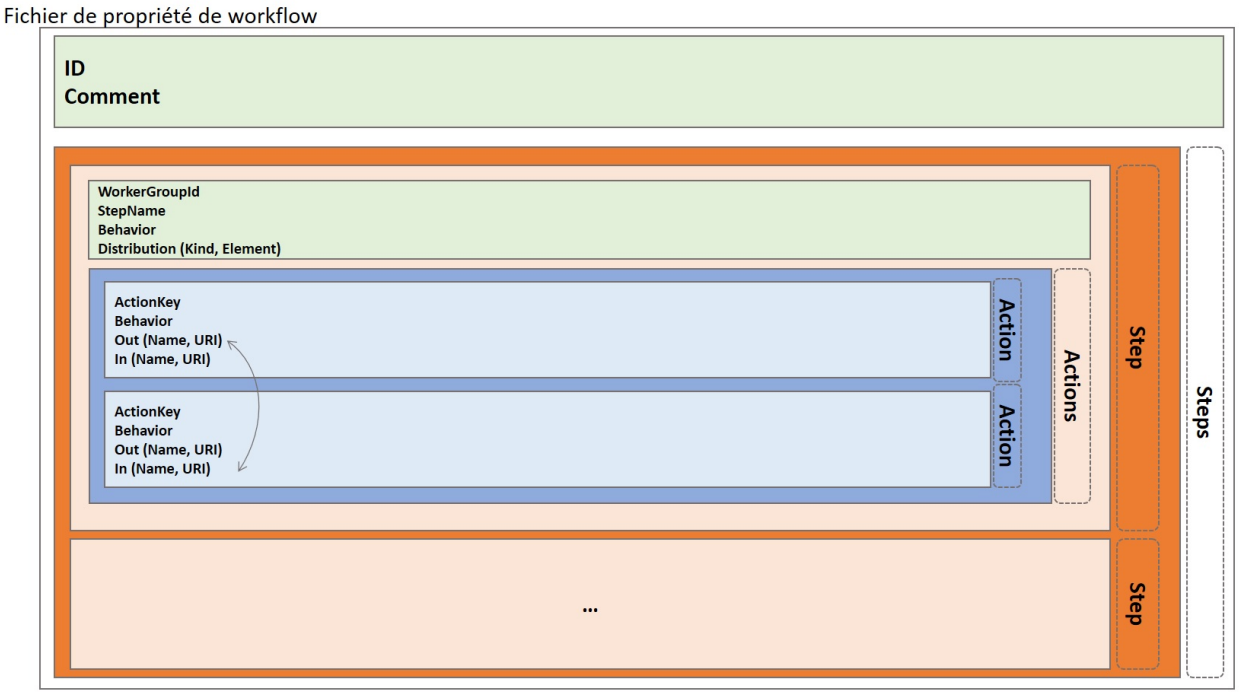

Fig. 1.2 – Structure du fichier de définition du workflow

Un Workflow est défini en JSON avec la structure suivante :

- un bloc en-tête contenant :
  - `ID` : identifiant unique du workflow,
  - `Comment` : description du workflow ou toutes autres informations utiles concernant le workflow
- une liste d'étapes dont la structure est la suivante :
  - `workerGroupId` : identifiant de famille de Workers,
  - `stepName` : nom de l'étape, servant de clé pour identifier l'étape,
  - `Behavior` : modèle d'exécution pouvant avoir les types suivants :
    - BLOCKING : le traitement est bloqué en cas d'erreur, il est nécessaire de recommencer le workflow,
    - NOBLOCKING : le traitement peut continuer malgrée les erreurs ou avertissements,
    - FINALLY : le traitement correspondant est toujours exécuté
  - `Distribution` : modèle de distribution, décrit comme suit :
    - `Kind` : un type pouvant être REF (i.e. élément unique) ou LIST (i.e. liste d'éléments)
    - `Element` : l'élément de distribution indiquant l'élément unique sous forme d'URI (REF) ou la liste d'éléments en pointant vers un dossier (LIST).
  - une liste d'Actions :
    - `ActionKey` : nom de l'action
    - `Behavior` : modèle d'exécution pouvant avoir les types suivants : - BLOCKING : l'action est bloquante en cas d'erreur. Les actions suivantes (de la meme étape) ne seront pas éxécutées. - NOBLOCKING : l'action peut continuer malgrée les erreurs ou avertissements.
    - `In` : liste de paramètres d'entrées : - `Name` : nom utilisé pour référencer cet élément entre différents handlers d'une même étape,

- `URI` : cible comportant un schema (WORKSPACE, MEMORY, VALUE) et un path où chaque handler peut accéder à ces valeurs via le handlerIO : - WORKSPACE : path indique le chemin relatif sur le workspace (implicitement un File), - MEMORY : path indique le nom de la clef de valeur (implicitement un objet mémoire déjà alloué par un Handler précédent), - VALUE : path indique la valeur statique en entrée (implicitement une valeur String).

- `Out` : liste de paramètres de sorties : - `Name` : nom utilisé pour référencer cet élément entre différents handlers d'une même étape,
  - `URI` : cible comportant un schema (WORKSPACE, MEMORY) et un path où chaque handler peut stocker les valeurs finales via le handlerIO : - WORKSPACE : path indique le chemin relatif sur le workspace (implicitement un File local), - MEMORY : path indique le nom de la clef de valeur (implicitement un objet mémoire).

Le code ci-dessous, à titre informatif, donne un exemple partiel de workflow, avec les notions étapes et actions.

```
{
  "id": "DefaultIngestWorkflow",
  "comment": "Default Ingest Workflow V6",
  "steps": [
    {
      "workerGroupId": "DefaultWorker",
      "stepName": "STP_INGEST_CONTROL_SIP",
      "behavior": "BLOCKING",
      "distribution": {
        "kind": "REF",
        "element": "SIP/manifest.xml"
      },
      "actions": [
        {
          "action": {
            "actionKey": "CHECK_SEDA",
            "behavior": "BLOCKING"
          }
        },
        {
          "action": {
            "actionKey": "CHECK_MANIFEST",
            "behavior": "BLOCKING",
            "out": [
              {
                "name": "mapsBDOtoOG.file",
                "uri": "WORKSPACE:Maps/OG_TO_ARCHIVE_ID_MAP.json"
              }
            ]
          }
        },
        {
          "action": {
            "actionKey": "CHECK_CONSISTENCY",
            "behavior": "NOBLOCKING",
            "in": [
              {
                "name": "mapsBDOtoOG.file",
                "uri": "WORKSPACE:Maps/OG_TO_ARCHIVE_ID_MAP.json"
              }
            ]
          }
        }
      ]
```

```
      }
   ]
}
```

1.2 Annexes

Table des figures

Liste des tableaux