Description des workflows et des opérations
Partie 1

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

État du document

◯ En projet ◯ Vérifié ◉ Validé

Maîtrise du document

Responsabilité	Nom	Entité	Date
Rédaction	Équipe Vitam	Équipe Vitam	09/01/2019
Vérification	MVI	Équipe Vitam	23/02/2021
Validation	AGR	Équipe Vitam	15/03/2021

Suivi des modifications

Version	Date	Auteur	Modifications
1.0	09/01/2019		Génération à partir de l'ancien document en RST
1.1	25/01/2019	NMO	Relecture
1.2	30/01/2019	EVA	Relecture
1.3	30/01/2019	MRE	Relecture
2.0	30/01/2019	MRE	Finalisation du document pour publication de la Release 9
2.1	07/02/2019	MVI	Mise à jour pour tenir compte des fonctionnalités mises en œuvre pendant la *Release 10 :* • chapitre 5, section III (« Workflow d'administration d'un référentiel des formats ») : précisions sur les cas WARNING (section III.A.1) ; précisions sur le contenu des champs suivants : EvDetDateTime, Warnings et UpdatedPUIDs. • chapitre 5, section XII (« Workflow d'administration d'un référentiel des griffons ») : corrections diverses : le champ Identifier fonctionne en mode VITAM esclave, le champ ExecutableVersion n'est pas un entier. • chapitre 5, section XIII (« Workflow d'administration d'un référentiel des scénarios de préservation ») : ajout de l'étape « Processus de génération du rapport d'import du référentiel des scénarios de préservation SCENARIO_REPORT » et d'une partie sur la « Structure du rapport d'administration du référentiel des scénarios de préservation ».
2.2	15/04/2019	MAF	Mise à jour et corrections diverses
3.0	24/04/2019	MRE	Finalisation pour publication Release 10
3.1	14/08/2019	MVI	Mise à jour pour tenir compte des fonctionnalités mises en œuvre pendant la *Release 11 :* • chapitre 2, section V (« Rapport de l'audit de cohérence ») : mise à jour du rapport ; • chapitre 11, section II (« Rapport de

			préservation ») : création de la section.
3.2	02/09/2019	JPP	Mise à jour pour tenir des fonctionnalités mises en œuvre pendant la Release 11 : • chapitre 8, section III et IV.
4.0	09/09/2019	MAF	Finalisation pour publication Release 11
4.1	24/09/2019	GFO	Modification multi-stratégies
4.2	23/10/2019	GFO	Mise à jour des Audits
4.3	19/11/2019	MAF	Mise à jour pour tenir des fonctionnalités mises en œuvre pendant la Release 12
5.0	29/11/2019	AGR	Finalisation pour publication Release 12
6.0	20/03/2020	AGR	Séparation du document en 2 parties et finalisation pour publication Release 13
6.1	24/06/2020	MVI	Mise à jour pour tenir des fonctionnalités mises en œuvre pendant la Release 14 : • chapitre 4, section 4.1 (« Workflow d'entrée ») : ajout des tâches CHECK_OBJECT_SIZE et CHECK_ATTACHEMENT ;
7.0	06/07/2020	AGR	Finalisation pour publication Release 14
7.1	26/10/2020	MVI	Mise à jour pour tenir des fonctionnalités mises en œuvre pendant la Release 15 : • chapitre 2, section 2.2 (« Workflow d'audit de vérification des journaux sécurisés ») : ajout de la section • chapitre 5, section 5.2.3 (« Structure du rapport du référentiel des règles de gestion ») : mise à jour de la modélisation du rapport.
8.0	26/10/2020	AGR	Finalisation pour publication Release 15
8.1	23/02/2021	MVI	Mise à jour pour tenir des fonctionnalités mises en œuvre pendant la Release 16 : • chapitre 4, section 4.1 (« Workflow d'entrée ») : ajout de la tâche de vérification de l'intégrité du bordereau de transfert (MANIFEST_DIGEST_CHECK) ; refonte du processus de mise à jour des groupes d'objets (STP_UPDATE_OBJECT_GROUP).
9.0	15/03/2021	AGR	Finalisation pour publication Release 16

Licence

Table des matières

1. INTRODUCTION

> ***Avertissement :*** *Cette documentation reflète l'état actuel de la solution logicielle Vitam. Elle est susceptible de changer dans les prochaines releases pour tenir compte des développements de la solution logicielle Vitam.*

1.1. Objectif du document

Ce document a pour objectif de présenter les différents processus employés par la solution logicielle Vitam. Il est destiné aux administrateurs aussi bien techniques que fonctionnels, aux archivistes souhaitant avoir une connaissance plus avancée du logiciel ainsi qu'aux développeurs.

Il explicite chaque processus (appelé également « workflow »), et pour chacun leurs tâches, traitements et actions.

Ce document comprend du matériel additionnel pour faciliter la compréhension des processus comme des fiches récapitulatives et des schémas. Il explique également la manière dont est formée la structure des fichiers de workflow.

Cette première partie du document décrit les workflows suivants :

- Audits
- Export DIP
- Ingest
- Masterdata

1.2. Description d'un processus

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

Pour chacun de ces éléments, le document décrit :

- La règle générale qui s'applique à cet élément,
- Les statuts de sortie possibles (OK, KO…), avec les raisons de ces sorties et les clés associées,
- Des informations complémentaires, selon le type d'élément traité.

Un « traitement » désigne ci-dessous une opération, une étape ou une tâche. Chaque traitement peut avoir à son issue un des statuts suivant :

- OK : le traitement s'est déroulé comme attendu et le système a été modifié en conséquence.
- Warning : le traitement a atteint son objectif mais le système émet une réserve. Soit :
  - Le système suspecte une anomalie lors du déroulement du traitement sans pouvoir le confirmer lui-même et lève une alerte à destination de l'utilisateur afin que celui-ci puisse valider qu'il s'agit du comportement souhaité.

    Exemple : un SIP versé sans objet provoque une opération en warning, car le fait de ne verser qu'une arborescence d'unités archivistiques sans aucun objet peut être suspect (au sens métier).
  - Le système a effectué un traitement entraînant une modification de données initialement non prévue par l'utilisateur.

    Exemple : la solution logicielle Vitam a détecté un format de fichier en contradiction avec le format décrit dans le bordereau de transfert. Elle enregistre alors ses propres valeurs en base de données au lieu de prendre celles du bordereau et utilise le warning pour en avertir l'utilisateur.
  - Le système a effectué un traitement dont seule une partie a entraîné une modification de données. L'autre partie de ce traitement s'est terminée en échec sans modification (KO).

Exemple : une modification de métadonnées en masse d'unités archivistiques dont une partie de la modification est OK et une partie est KO : le statut de l'étape et de l'opération sera Warning.

- KO : le traitement s'est terminé en échec et le système n'a pas été modifié en dehors des éléments de traçabilités tels que les journaux et les logs. L'intégralité du traitement pourrait être rejouée sans provoquer l'insertion de doublons.
- Fatal : le traitement s'est terminé en échec a cause d'un problème technique. L'état du système dépend de la nature du traitement en fatal et une intervention humaine est requise pour expertiser et résoudre la situation. Lorsque le statut FATAL survient à l'intérieur d'une étape (par exemple dans une des tâches ou une des actions de l'étape), c'est toute l'étape qui est mise en pause. Si cette étape est rejouée, les objets déjà traités avant le problème technique ne sont pas traités à nouveau : le workflow reprend exactement là où il s'était arrêté et commence par rejouer l'action sur l'objet qui a provoqué l'erreur.

Un workflow peut être terminé, en cours d'exécution ou être en pause. Un workflow en pause représente le processus arrêté à une étape donnée. Chaque étape peut être mise en pause : ce choix dépend du mode de versement (le mode pas à pas marque une pause à chaque étape), ou du statut (le statut FATAL met l'étape en pause). Les workflows en pause sont visibles dans l'IHM dans l'écran « Gestion des opérations ».

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

- Bloquant
  - Si une action bloquante est identifiée en erreur, le workflow est alors arrêté en erreur. Seules les actions nécessaire à l'arrêt du workflow sont alors exécutées.
- Non bloquant
  - Si une action non bloquante est identifiée en erreur, elle seule sera en erreur et le workflow continuera normalement.

1.3.Structure d'un fichier Properties du Workflow

Les fichiers **Properties** (par exemple *DefaultIngestWorkflow.json*) permettent de définir la structure du Workflow pour les étapes, tâches et traitements réalisés dans le module d'Ingest Interne, en excluant les étapes et traitements réalisés dans le module d'Ingest externe.

Un Workflow est défini en JSON avec la structure suivante :

- un bloc en-tête contenant :
  - `ID` : identifiant unique du workflow,
  - `Identifier` : clé du workflow,
  - `Name` : nom du workflow,
  - `TypeProc` : catégorie 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 à la tâche en erreur. Les étapes FINALLY (définition ci-dessous) sont tout de même exécutées,
    - NOBLOCKING : le traitement peut continuer malgré les éventuels erreurs ou avertissements,
    - FINALLY : le traitement correspondant est toujours exécuté, même si les étapes précédentes se sont terminées en échec.
  - `Distribution` : modèle de distribution, décrit comme suit :
    - `Kind` : un type pouvant être REF (un élément unique) ou LIST (une liste d'éléments hiérarchisés) ou encore LIST_IN_FILE (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),
    - `Type` : le type des objets traités (ObjectGroup uniquement pour le moment),
    - `StatusOnEmptyDistribution` : permet dans le cas d'un traitement d'une liste vide, de surcharger le statut WARNING par un statut prédéfini,
    - `BulkSize`: taille de la liste, valeur à spécifier ex: « bulkSize » : 1000. La valeur par défaut est de 16.
- 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 même étape) ne seront pas exécutées,
        - NOBLOCKING : l'action peut continuer malgré les éventuels erreurs ou avertissements.
    - `LifecycleLog`: action indiquant le calcul sur les LFC. Valeur du champ « DISABLED ».
    - `In` : liste de paramètres d'entrées :
        - `Name` : nom utilisé pour référencer cet élément entre différents handlers d'une même étape,
        - `URI` : cible comportant un schéma (WORKSPACE, MEMORY, VALUE) et un path où chaque handler peut accéder à ces valeurs via le handlerIO :
            - WORKSPACE : path indiquant le chemin relatif sur le workspace (implicitement un File),
            - MEMORY : path indiquant le nom de la clef de valeur (implicitement un objet mémoire déjà alloué par un handler précédent),
            - VALUE : path indiquant 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 schéma (WORKSPACE, MEMORY) et un path où chaque handler peut stocker les valeurs finales via le handlerIO :
        - WORKSPACE : path indiquant le chemin relatif sur le workspace (implicitement un File local),
        - MEMORY : path indiquant le nom de la clé de valeur (implicitement un objet mémoire).

Bloc d'en tête

Nom de l'étape

Liste des tâches de l'étape

```
{
  "id": "DEFAULT_WORKFLOW",
  "name": "Default Ingest Workflow",
  "identifier": "PROCESS_SIP_UNITARY",
  "typeProc": "INGEST",
  "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_HEADER",
            "behavior": "BLOCKING",
            "in": [
              {
                "name": "checkContract",
                "uri": "VALUE:true"
              },
              {
                "name": "checkOriginatingAgency",
                "uri": "VALUE:true"
              },
              {
                "name": "checkProfile",
                "uri": "VALUE:true"
              }
```

Illustration 1: Exemple d'un fichier properties - Ingest Interne

2. AUDIT

2.1. Workflow de contrôle d'intégrité d'un journal sécurisé

Cette section décrit le processus (workflow) de contrôle d'intégrité d'un journal sécurisé mis en place dans la solution logicielle Vitam.

Celui-ci est défini dans le fichier *"DefaultCheckTraceability.json"* (situé ici : sources/processing/processing-management/src/main/resources/workflows).

Toutes les étapes, tâches et traitements sont journalisés dans le journal des opérations, et décrivent le processus (clé et description de la clé associée dans le journal des opérations) tel qu'implémenté dans la version actuelle de la solution logicielle Vitam.

2.1.1. Processus de contrôle d'intégrité d'un journal sécurisé (vision métier)

Le processus de contrôle d'intégrité débute lorsqu'un identifiant d'opération de sécurisation des journaux d'opération, des journaux du cycle de vie ou du journal des écritures est soumis au service de contrôle d'intégrité des journaux sécurisés. Le service permet de récupérer le journal sécurisé, d'extraire son contenu et de valider que son contenu n'a pas été altéré.

Pour cela, le service calcule un arbre de Merkle à partir des journaux d'opérations que contient le journal sécurisé, puis en calcule un second à partir des journaux correspondants disponibles dans la solution logicielle Vitam. Une comparaison est ensuite effectuée entre ces deux arbres et celui contenu dans les métadonnées du journal sécurisé.

Ensuite, dans une dernière étape, le tampon d'horodatage est vérifié et validé.

2.1.2. Processus de préparation de la vérification des journaux sécurisés (STP_PREPARE_TRACEABILITY_CHECK)

Préparation de la vérification des journaux sécurisés PREPARE_TRACEABILITY_CHECK (PrepareTraceabilityCheckProcessActionHandler.java)

- **Règle** : tâche permettant de vérifier que l'opération donnée en entrée est de type TRACEABILITY et à récupérer le fichier au format .zip associé à cette opération et à extraire son contenu
- **Type** : bloquant
- **Statuts** :
  - OK : l'opération donnée en entrée est une opération de type TRACEABILITY, le zip a été trouvé et son contenu extrait (PREPARE_TRACEABILITY_CHECK.OK = Succès de la préparation de la vérification des journaux sécurisés)
  - KO : l'opération donnée en entrée n'est pas une opération de type TRACEABILITY (PREPARE_TRACEABILITY_CHECK.KO = Échec de la préparation de la vérification des journaux sécurisés)
  - FATAL : une erreur technique est survenue lors du processus de préparation de vérification (PREPARE_TRACEABILITY_CHECK.FATAL = Erreur technique lors de la préparation de la vérification des journaux sécurisés)

2.1.3.Processus de vérification de l'arbre de Merkle (STP_MERKLE_TREE)

2.1.3.1. Vérification de l'arbre de Merkle CHECK_MERKLE_TREE (VerifyMerkleTreeActionHandler.java)

- **Règle** : tâche consistant à recalculer l'arbre de Merkle des journaux contenus dans le journal sécurisé, à calculer un autre arbre à partir des journaux indexés correspondants et à vérifier que tous deux correspondent à celui stocké dans les métadonnées du journal sécurisé
- **Type** : bloquant
- **Statuts** :
  - OK : les arbres de Merkle correspondent (CHECK_MERKLE_TREE.OK = Succès de la vérification de l'arbre de MERKLE)
  - KO : les arbres de Merkle ne correspondent pas (CHECK_MERKLE_TREE.KO = Échec de la vérification de l'arbre de MERKLE)
  - FATAL : une erreur technique est survenue lors de la vérification des arbres de Merkle (CHECK_MERKLE_TREE.FATAL = Erreur technique lors de la vérification de l'arbre de MERKLE)

La tâche Check_Merkle_Tree contient les traitements suivants :

2.1.3.2. Comparaison de l'arbre de Merkle avec le Hash enregistré CHECK_MERKLE_TREE.COMPARE_MERKLE_HASH_WITH_SAVED_HASH

- **Règle** : traitement consistant à vérifier que l'arbre de Merkle calculé à partir des journaux contenus dans le journal sécurisé est identique à celui stocké dans les métadonnées du journal sécurisé
- **Type** : bloquant
- **Statuts** :
  - OK : l'arbre de Merkle des journaux contenus dans le journal sécurisé correspond à celui stocké dans les métadonnées du journal sécurisé (CHECK_MERKLE_TREE.COMPARE_MERKLE_HASH_WITH_SAVED_HASH.OK = Succès de la comparaison de l'arbre de MERKLE avec le Hash enregistré)
  - KO : l'arbre de Merkle des journaux contenus dans le journal sécurisé ne correspond pas à celui stocké dans les métadonnées du journal sécurisé (CHECK_MERKLE_TREE.COMPARE_MERKLE_HASH_WITH_SAVED_HASH.KO = Échec de la comparaison de l'arbre de MERKLE avec le Hash enregistré)
  - FATAL : une erreur technique est survenue lors de la comparaison de l'arbre de MERKLE avec le Hash enregistré (CHECK_MERKLE_TREE.COMPARE_MERKLE_HASH_WITH_SAVED_HASH.FATAL = Erreur technique lors de la comparaison de l'arbre de MERKLE avec le Hash enregistré)

2.1.3.3. Comparaison de l'arbre de Merkle avec le Hash indexé CHECK_MERKLE_TREE.COMPARE_MERKLE_HASH_WITH_INDEXED_HASH

- **Règle** : traitement consistant à vérifier que l'arbre de Merkle calculé à partir des journaux indexés est identique à celui stocké dans les métadonnées du journal sécurisé
- **Type** : bloquant
- **Statuts** :
  - OK : l'arbre de Merkle des journaux indexés correspond à celui stocké dans les métadonnées du journal sécurisé (CHECK_MERKLE_TREE.COMPARE_MERKLE_HASH_WITH_INDEXED_HASH.OK = Succès de la comparaison de l'arbre de MERKLE avec le Hash indexé)
  - KO : l'arbre de Merkle des journaux indexés ne correspond pas à celui stocké dans les métadonnées du journal sécurisé

(CHECK_MERKLE_TREE.COMPARE_MERKLE_HASH_WITH_INDEXED_HASH.KO = Échec de la comparaison de l'arbre de MERKLE avec le Hash indexé)

- FATAL : une erreur technique est survenue lors de la comparaison de l'arbre de Merkle des journaux indexés avec celui stocké dans les métadonnées du journal sécurisé (CHECK_MERKLE_TREE.COMPARE_MERKLE_HASH_WITH_INDEXED_HASH.FATAL = Erreur technique lors de la comparaison de l'arbre de MERKLE avec le Hash indexé)

2.1.4.Processus de vérification de l'horodatage (STP_VERIFY_STAMP)

2.1.4.1. Vérification et validation du tampon d'horodatage VERIFY_TIMESTAMP (VerifyTimeStampActionHandler.java)

- **Règle** : tâche consistant à vérifier et à valider le tampon d'horodatage
- **Type** : bloquant
- **Statuts** :
  - OK : le tampon d'horadatage est correct (VERIFY_TIMESTAMP.OK = Succès de la vérification de l'horodatage)
  - KO : le tampon d'horadatage est incorrect (VERIFY_TIMESTAMP.KO = Échec de la vérification de l'horodatage)
  - FATAL : une erreur technique est survenue lors de la vérification du tampon d'horodatage (VERIFY_TIMESTAMP.FATAL = Erreur technique lors de la vérification de l'horodatage)

La tâche Verify_Timestamp contient les traitements suivants :

2.1.4.2. Comparaison du tampon du fichier (token.tsp) par rapport au tampon enregistré dans le logbook VERIFY_TIMESTAMP.COMPARE_TOKEN_TIMESTAMP

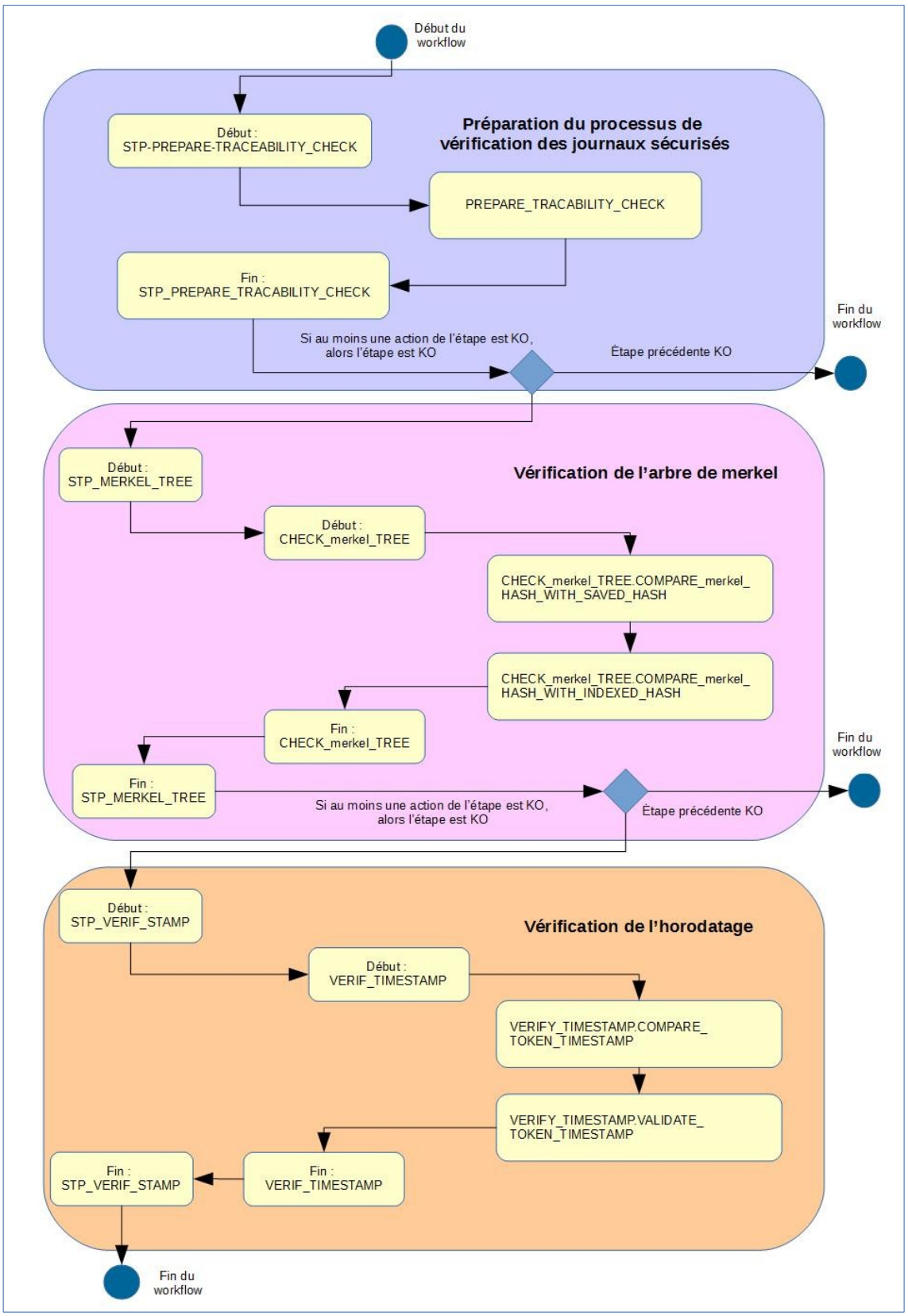
- **Règle** : traitement consistant à vérifier que le tampon enregistré dans la collection logbookOperation est le même que celui présent dans le fichier zip généré
- **Type** : bloquant
- **Statuts** :
  - OK : les tampons sont identiques (VERIFY_TIMESTAMP.COMPARE_TOKEN_TIMESTAMP.OK = Succès de la comparaison des tampons d'horodatage)
  - KO : les tampons sont différents (VERIFY_TIMESTAMP.COMPARE_TOKEN_TIMESTAMP.KO = Échec de la comparaison des tampons d'horodatage)
  - FATAL : erreur technique lors de la vérification des tampons (VERIFY_TIMESTAMP.COMPARE_TOKEN_TIMESTAMP.FATAL = Erreur technique lors de la comparaison des tampons d'horodatage)

2.1.4.3. Validation du tampon d'horodatage VERIFY_TIMESTAMP.VALIDATE_TOKEN_TIMESTAMP

- **Règle** : traitement consistant à vérifier cryptographiquement le tampon et à vérifier la chaîne de certification
- **Type** : bloquant
- **Statuts** :
  - OK : le tampon est validé (VERIFY_TIMESTAMP.VALIDATE_TOKEN_TIMESTAMP.OK = Succès de la validation du tampon d'horodatage)
  - KO : le tampon est invalidé (VERIFY_TIMESTAMP.VALIDATE_TOKEN_TIMESTAMP.KO = Échec de la validation du tampon d'horodatage)
  - FATAL : erreur technique lors de la validation du tampon d'horodatage (VERIFY_TIMESTAMP.VALIDATE_TOKEN_TIMESTAMP.FATAL = Erreur technique lors de la validation du tampon d'horodatage)

2.1.5.Structure du workflow de contrôle d'intégrité d'un journal sécurisé

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



2.2.Workflow d'audit de vérification des journaux sécurisés

Cette section décrit le processus (workflow) d'audit de vérification des sécurisations en masse mis en place afin de s'assurer de la non perte et de la validité des donnés conservées dans la solution logicielle Vitam.

Celui-ci est défini dans le fichier « LinkedCheckTraceability.json » (situé ici : sources/processing/processing-management/src/main/resources/workflows/LinkedCheckTraceability.json)

Toutes les étapes, tâches et traitements sont journalisés dans le journal des opérations, et décrivent le processus (clé et description de la clé associée dans le journal des opérations) tel qu'implémenté dans la version actuelle de la solution logicielle Vitam.

2.2.1.Processus d'audit de vérification des journaux sécurisés (vision métier)

L'audit de sécurisation doit permettre de vérifier que les journaux de sécurisation existent, pour chaque type de journal, sur une période définie, et être effectué sur chaque offre selon la stratégie par défaut.

2.2.2.Processus de préparation de la vérification des journaux sécurisés (STP_PREPARE_TRACEABILITY_LINKED_CHECK)

Règle : Étape qui permet de retrouver la liste des journaux de sécurisation tels que définis dans la requête DSL (type de journal et intervalle de temps) et de récupérer la liste des stratégies de stockage.

Type : Bloquant

Statuts :

- **OK** : Tous les journaux demandés par la requête ont bien été récupérés et les stratégies également (PREPARE_TRACEABILITY_CHECK.OK = Succès de la préparation de la vérification des journaux sécurisés)
- **FATAL** : une erreur technique est survenue lors du processus de préparation de vérification (PREPARE_TRACEABILITY_CHECK.FATAL = Erreur technique lors de la préparation de la vérification des journaux sécurisés)
- **WARNING** : Avertissement lors du processus de préparation de la vérification des journaux sécurisés (STP_PREPARE_TRACEABILITY_CHECK.WARNING =Avertissement lors du processus de préparation de la vérification des journaux sécurisés)

2.2.2.1. Préparation de la vérification des journaux sécurisés TRACEABILITY_LINKED_CHECK_PREPARE (TraceabilityLinkedCheckPreparePlugin.java)

- **Règle :** Tâche permettant de retrouver la liste des journaux de sécurisation tels que définis dans la requête DSL (type de journal et intervalle de temps).
- **Type :** Bloquant
- **Statuts** :
    - OK : Tous les journaux demandés par la requête ont bien été récupérés (TRACEABILITY_LINKED_CHECK_PREPARE.OK= Succès de la préparation de la vérification des journaux sécurisés)

- FATAL : Erreur technique dans la récupération des journaux demandés (TRACEABILITY_LINKED_CHECK_PREPARE.FATAL= Erreur technique lors de la préparation de la vérification des journaux sécurisés)
- WARNING : Pas de journaux trouvés ou certains journaux ne possède pas de données (TRACEABILITY_LINKED_CHECK_PREPARE.WARNING= Avertissement lors de la préparation de la vérification des journaux sécurisés)

2.2.2.2. Récupération de la liste des stratégies de stockage PREPARE_STORAGE_STRATEGIES

(PrepareStorageStrategiesPlugin.java)

- **Règle :** Tâche consistant à récupérer la configuration des stratégies de stockage définies pour la plateforme.
- **Type :** Bloquant
- **Statuts :**
  - OK : Les données ont été récupérées et enregistrées dans un fichier de travail avec succès (PREPARE_STORAGE_STRATEGIES.OK=Succès de la récupération de la liste des stratégies de stockage)
  - FATAL : Erreur technique lors de la récupération de la liste des stratégies de stockage (PREPARE_STORAGE_STRATEGIES.FATAL=Erreur technique lors de la récupération de la liste des stratégies de stockage)

2.2.3.Processus de la vérification des journaux sécurisés (STP_TRACEABILITY_LINKED_CHECKS)

Règle : Étape permettant de vérifier que les journaux de sécurisation sont bien sur toutes les offres attendues et de vérifier leur intégrité.

Type : Non bloquant

Statuts :

- OK : Tous les journaux ont été trouvés sur les offres définies par la stratégie par défaut et leur intégrité est bien vérifiée (STP_TRACEABILITY_LINKED_CHECKS.OK=Succès du processus de la vérification des journaux sécurisés)
- KO : Tous les journaux n'ont été trouvés sur les offres définies par la stratégie par défaut OU leur intégrité n'est vérifiée (STP_TRACEABILITY_LINKED_CHECKS.KO=Échec du processus de la vérification des journaux sécurisés)
- FATAL : Erreur technique lors de la vérification du stockage et de l'intégrité des journaux de sécurisation (STP_TRACEABILITY_LINKED_CHECKS.FATAL=Erreur technique lors du processus de la vérification des journaux sécurisés)
- WARNING : Aucun journal de sécurisation correspondant à la requête (type de journal et intervalle de temps) n'a été trouvé (STP_TRACEABILITY_LINKED_CHECKS.WARNING=Avertissement lors du processus de la vérification des journaux sécurisés)

2.2.3.1. Vérification d'existence de fichier de sécurisation dans les offres de stockage et récupération du hash correspondant RETRIEVE_SECURE_TRACEABILITY_DATA_FILE

(RetrieveSecureTraceabilityDataFilePlugin.java)

- **Règle :** Tâche de vérification de l'existence des fichiers de sécurisation dans les offres de stockage par défaut et récupération du hash correspondant.
- **Type :** Non bloquant
- **Statuts :**
  - OK : Succès de la vérification d'existence de fichier de sécurisation dans les offres de stockage et récupération du hash correspondant (RETRIEVE_SECURE_TRACEABILITY_DATA_FILE.OK=Succès de la vérification d"existence de fichier de sécurisation dans les offres de stockage et récupération du hash correspondant)
  - KO : Au moins un fichier sécurisation n'a pas été trouvé sur l'une des offres de stockage (RETRIEVE_SECURE_TRACEABILITY_DATA_FILE.KO=Échec de la vérification d"existence de fichier de sécurisation dans les offres de stockage et récupération du hash correspondant
  - FATAL : Erreur technique sur au moins une offre au moment de la vérification de l'existence des fichiers de sécurisation(RETRIEVE_SECURE_TRACEABILITY_DATA_FILE.FATAL=Erreur technique lors de la vérification d"existence de fichier de sécurisation dans les offres de stockage)

2.2.3.2. Vérification de l'intégrité du fichier de sécurisation CHECKS_SECURE_TRACEABILITY_DATA_HASHES

(ChecksSecureTraceabilityDataHashesPlugin.java)

- **Règle :** Comparaison du hash du fichier de sécurisation présent dans l'offre référente avec les hash des fichiers de sécurisation obtenus par la tâche précédente.
- **Type :** Non bloquant
- **Statuts :**
  - OK : L'intégrité du fichier de sécurisation est vérifiée (CHECKS_SECURE_TRACEABILITY_DATA_HASHES.OK=Succès de la vérification de l"intégrité du fichier de sécurisation)
  - KO : L'intégrité du fichier de sécurisation est n'est pas vérifiée (CHECKS_SECURE_TRACEABILITY_DATA_HASHES.KO=Échec de la vérification de l"intégrité du fichier de sécurisation)
  - FATAL : Erreur technique lors du contrôle de l'intégrité du fichier de sécurisation (CHECKS_SECURE_TRACEABILITY_DATA_HASHES.FATAL=Erreur technique lors de la vérification de l"intégrité du fichier de sécurisation)

2.2.3.3. Récupération du fichier de sécurisation EXTRACT_SECURE_TRACEABILITY_DATA_FILE

(ExtractSecureTraceabilityDataFilePlugin.java)

- **Règle :** Récupération et décompression du contenu du fichier compressé de sécurisation en vue de son traitement.
- **Type :** Non bloquant
- **Statuts :**
  - OK : Succès de la décompression du fichier de sécurisation (EXTRACT_SECURE_TRACEABILITY_DATA_FILE.OK=Succès de l''extraction du fichier de sécurisation)
  - FATAL : Erreur technique au moment de la décompression du fichier de sécurisation (EXTRACT_SECURE_TRACEABILITY_DATA_FILE.FATAL=Erreur technique lors de l''extraction du fichier de sécurisation)

2.2.3.4. Vérification de l'arbre de MERKLE CHECK_MERKLE_TREE

- **Règle :** Tâche consistant à recalculer l’arbre de Merkle des journaux contenus dans le journal sécurisé, à calculer un autre arbre à partir des journaux indexés correspondants et à vérifier que tous deux correspondent à celui stocké dans les métadonnées du journal sécurisé.
- **Type :** Non bloquant
- **Statuts :**
  - OK : les arbres de Merkle correspondent (CHECK_MERKLE_TREE.OK=Succès de la vérification de l''arbre de MERKLE)
  - KO : les arbres de Merkle ne correspondent pas (CHECK_MERKLE_TREE.KO=Échec de la vérification de l''arbre de MERKLE)
  - FATAL : une erreur technique est survenue lors de la vérification des arbres de Merkle (CHECK_MERKLE_TREE.FATAL=Erreur technique lors de la vérification de l''arbre de MERKLE)

Cette tache contient les traitements suivants :

2.2.3.5. 1. Comparaison de l’arbre de Merkle avec le hash enregistré

CHECK_MERKLE_TREE.COMPARE_MERKLE_HASH_WITH_SAVED_HASH

- Règle : Traitement consistant à vérifier que l’arbre de Merkle calculé à partir des journaux contenus dans le journal sécurisé est identique à celui stocké dans le fichier de sécurisation.
- Type : Non bloquant

- Statuts :

  - OK : L'arbre de Merkle des journaux contenus dans le journal sécurisé correspond à celui stocké dans le fichier de sécurisation. (CHECK_MERKLE_TREE.COMPARE_MERKLE_HASH_WITH_SAVED_HASH.OK=Succès de la comparaison de l''arbre de MERKLE avec le Hash enregistré)

  - KO : L'arbre de Merkle des journaux contenus dans le journal sécurisé ne correspond pas à celui stocké dans le fichier de sécurisation. (CHECK_MERKLE_TREE.COMPARE_MERKLE_HASH_WITH_SAVED_HASH.KO=Échec de la comparaison de l''arbre de MERKLE avec le Hash enregistré)

  - FATAL : Une erreur technique est survenue lors de la comparaison de l'arbre de MERKLE avec le hash enregistré (CHECK_MERKLE_TREE.COMPARE_MERKLE_HASH_WITH_SAVED_HASH.FATAL=Erreur technique lors de la comparaison de l''arbre de MERKLE avec le Hash enregistré)

2.2.3.6. 2. Comparaison de l'arbre de Merkle avec le hash indexé

CHECK_MERKLE_TREE.COMPARE_MERKLE_HASH_WITH_INDEXED_HASH

- **Règle :** Traitement consistant à vérifier que l'arbre de Merkle calculé à partir des journaux contenus dans le journal sécurisé est identique à celui stocké dans les métadonnées du journal sécurisé.

- **Type :** Non bloquant

- **Statuts :**

  - OK : l'arbre de Merkle des journaux indexés correspond à celui stocké dans les métadonnées du journal sécurisé (CHECK_MERKLE_TREE.COMPARE_MERKLE_HASH_WITH_INDEXED_HASH.OK =Succès de la comparaison de l''arbre de MERKLE avec le Hash enregistré)

  - KO : l'arbre de Merkle des journaux indexés correspond à celui stocké dans les métadonnées du journal sécurisé (CHECK_MERKLE_TREE.COMPARE_MERKLE_HASH_WITH_INDEXED_HASH.KO =Échec de la comparaison de l''arbre de MERKLE avec le Hash enregistré)

  - FATAL : une erreur technique est survenue lors de la comparaison de l'arbre de Merkle des journaux indexés avec celui stocké dans les métadonnées du journal sécurisé (CHECK_MERKLE_TREE.COMPARE_MERKLE_HASH_WITH_INDEXED_HASH.FATAL =Erreur technique lors de la comparaison de l''arbre de MERKLE avec le Hash enregistré)

2.2.3.7. Vérification des journaux des écritures (journal des offres de stockage) CHECKS_SECURE_TRACEABILITY_DATA_STORAGELOG

- **Règle :** Ce traitement vérifie que le (ou les) journaux des écritures sur les offres de stockage existent et sont intègres.

- **Type :** Bloquant

- **Statuts :**

  - OK : Le ou les journaux des offres de stockage existent et leur intégrité est vérifiée

(CHECKS_SECURE_TRACEABILITY_DATA_STORAGELOG.OK=Succès de la vérification des journaux des écritures)

- KO : Le ou les journaux des offres de stockage n'existent pas et/ou leur intégrité est n'est pas vérifiée (CHECKS_SECURE_TRACEABILITY_DATA_STORAGELOG.KO=Échec de la vérification des journaux des écritures)
- FATAL : Erreur technique lots de la vérification des journaux des écritures sur les offres de stockage (CHECKS_SECURE_TRACEABILITY_DATA_STORAGELOG.FATAL=Erreur technique lors de la vérification des journaux des écritures)

2.2.4. Processus de la finalisation de la vérification des journaux sécurisés STP_FINALIZE_TRACEABILITY_LINKED_CHECKS

Règle : Étape d'utilisation des résultats d'analyse et création du rapport d'audit des sécurisations

Type : Bloquant

Statuts :

- OK : Le rapport d'audit des sécurisations a été généré avec succès (STP_FINALIZE_TRACEABILITY_LINKED_CHECKS.KO=Échec du processus de la finalisation de la vérification des journaux sécurisés)
- FATAL : Erreur technique lors de la génération du rapport d'audit des sécurisations (STP_FINALIZE_TRACEABILITY_LINKED_CHECKS.FATAL=Erreur technique lors du processus de la finalisation de la vérification des journaux sécurisés)

2.2.4.1. Finalisation de la vérification des journaux sécurisés

TRACEABILITY_FINALIZATION

(TraceabilityFinalizationPlugin.java)

- **Règle :** Utilisation des résultats d'analyse et création du rapport d'audit des sécurisations.
- **Type :** Bloquant
- **Statuts :**
  - OK : Le rapport d'audit des sécurisations a été généré avec succès (TRACEABILITY_FINALIZATION.OK=Succès de la finalisation de la vérification des journaux sécurisés)
  - FATAL : Erreur technique lors de la génération du rapport d'audit des sécurisations (TRACEABILITY_FINALIZATION.FATAL=Erreur technique lors de la finalisation de la vérification des journaux sécurisés)

2.2.5.Audit de la vérification des journaux sécurisés LINKED_CHECK_SECURISATION

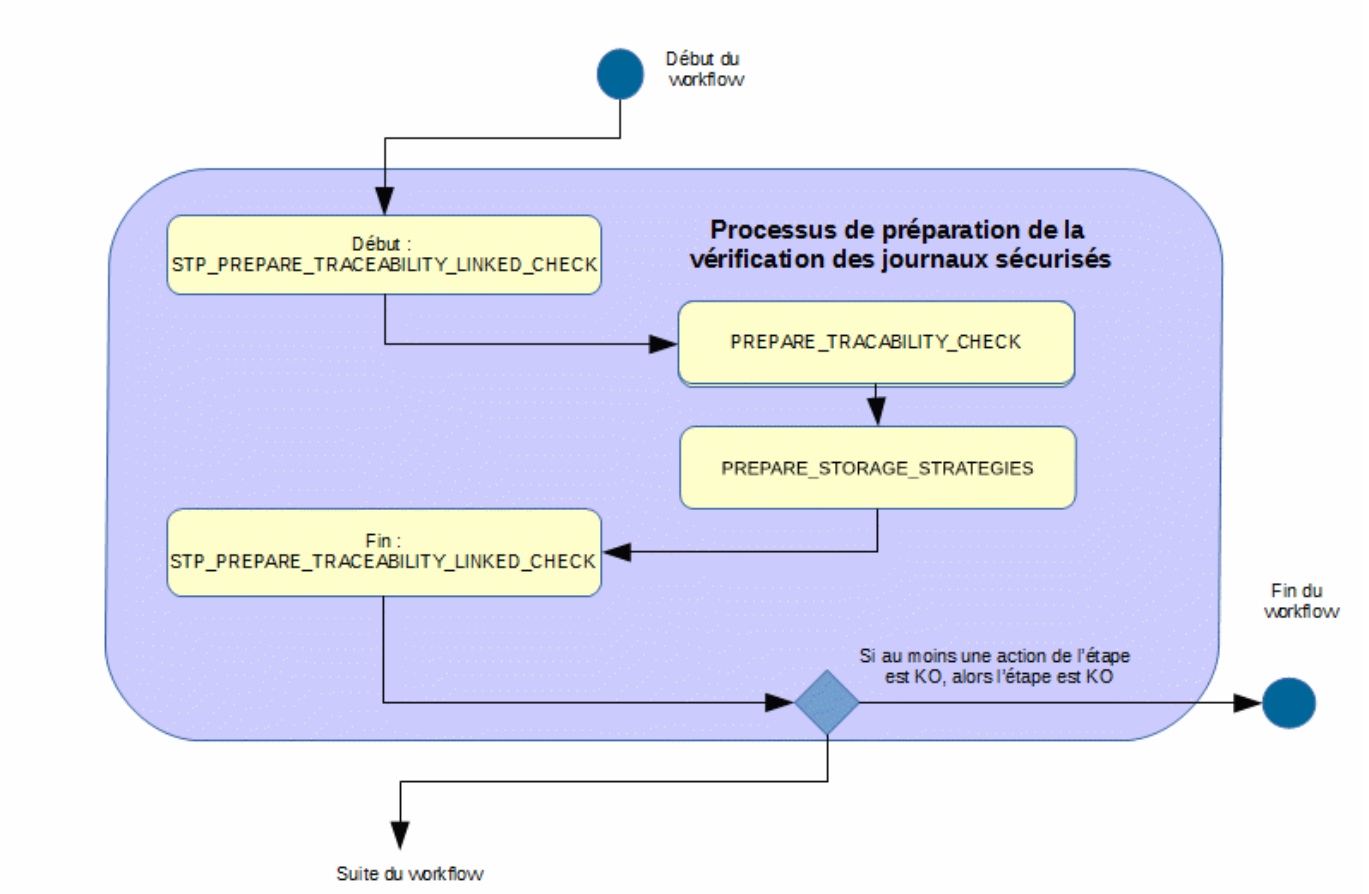
Règle : Affichage du résultat du workflow d'audit des sécurisations dans l'IHM démo

Type : N/A

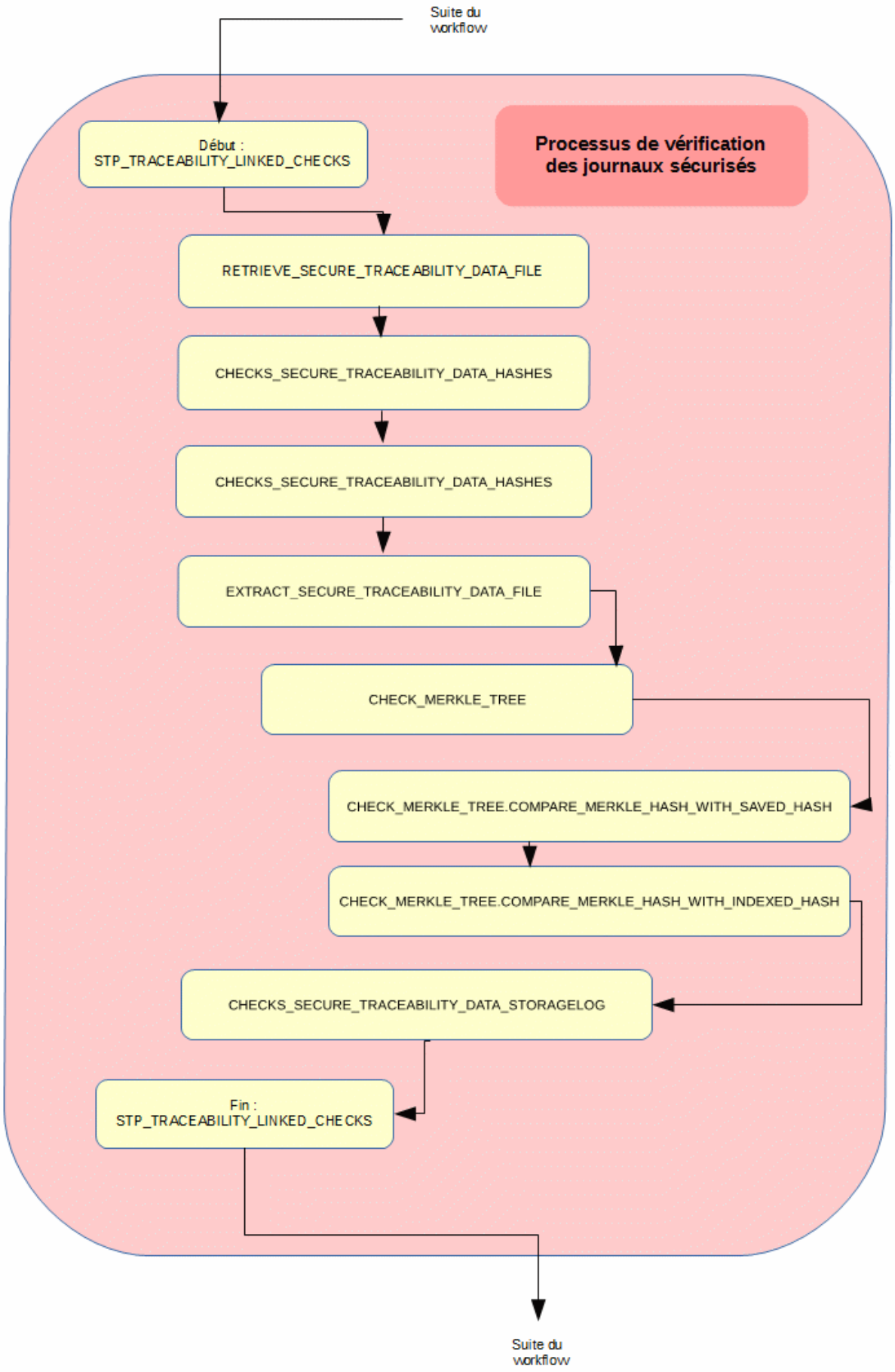
Statuts :

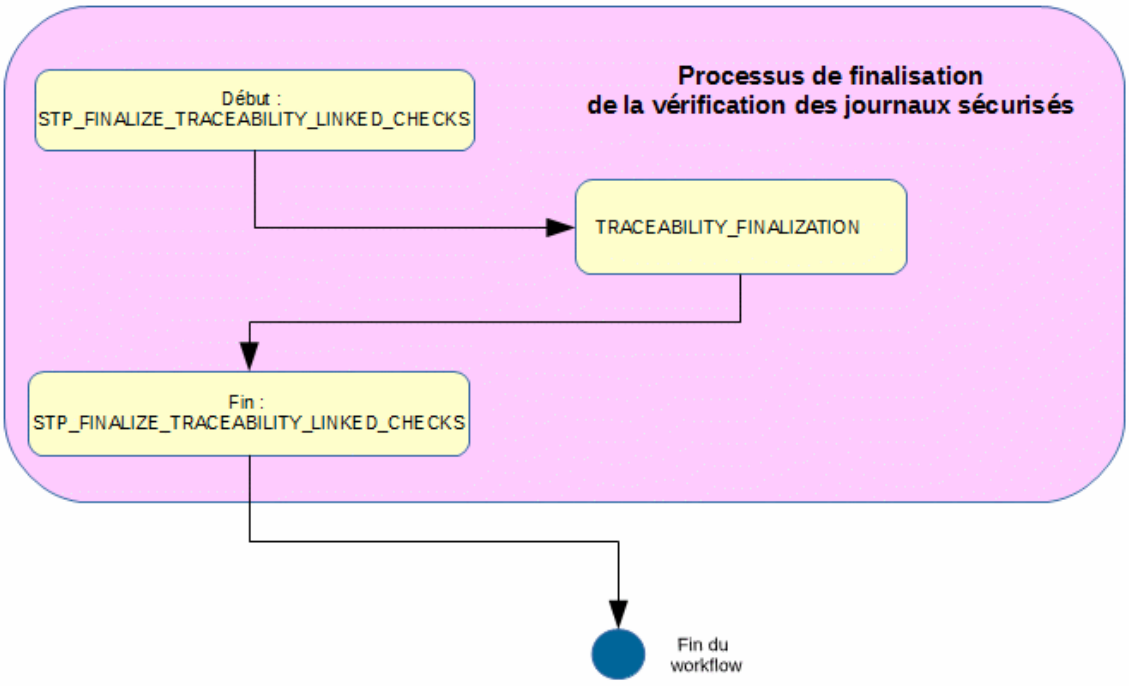
- OK : Le résultat global de l'audit des sécurisations est conforme à l'attendu (LINKED_CHECK_SECURISATION.OK=Succès d''audit de la vérification des journaux sécurisés)
- KO : Le résultat global de l'audit des sécurisations n'est pas conforme à l'attendu (LINKED_CHECK_SECURISATION.KO=Échec d''audit de la vérification des journaux sécurisés)
- FATAL : Erreur fatal lors de l'exécution du workflow d'audit des sécurisations (LINKED_CHECK_SECURISATION.FATAL=Erreur technique lors d''audit de la vérification des journaux sécurisés)
- WARNING : Le résultat du workflow d'audit des sécurisations est en avertissement (LINKED_CHECK_SECURISATION.WARNING=Avertissement lors d''audit de la vérification des journaux sécurisés)

2.2.6. Structure du workflow



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


Suite du workflow
Début : STP_TRACEABILITY_LINKED_CHECKS
Processus de vérification des journaux sécurisés
RETRIEVE_SECURE_TRACEABILITY_DATA_FILE
CHECKS_SECURE_TRACEABILITY_DATA_HASHES
CHECKS_SECURE_TRACEABILITY_DATA_HASHES
EXTRACT_SECURE_TRACEABILITY_DATA_FILE
CHECK_MERKLE_TREE
CHECK_MERKLE_TREE.COMPARE_MERKLE_HASH_WITH_SAVED_HASH
CHECK_MERKLE_TREE.COMPARE_MERKLE_HASH_WITH_INDEXED_HASH
CHECKS_SECURE_TRACEABILITY_DATA_STORAGELOG
Fin : STP_TRACEABILITY_LINKED_CHECKS
Suite du workflow


Processus de finalisation
de la vérification des journaux sécurisés
Début :
STP_FINALIZE_TRACEABILITY_LINKED_CHECKS
TRACEABILITY_FINALIZATION
Fin :
STP_FINALIZE_TRACEABILITY_LINKED_CHECKS
Fin du
workflow

2.3.Workflow de l'audit de l'existence et de l'intégrité des fichiers

Cette section décrit le processus (workflow) d'audit de l'existence et de l'intégrité des fichiers mis en place dans la solution logicielle Vitam.

Celui-ci est défini dans le fichier *« DefaultAuditWorkflow.json »* (situé ici : sources/processing/processing-management/src/main/resources/workflows).

Toutes les étapes, tâches et traitements sont journalisés dans le journal des opérations et décrivent le processus (clé et description de la clé associée dans le journal des opérations) tel qu'implémenté dans la version actuelle de la solution logicielle Vitam.

Le processus d'audit prend comme point d'entrée l'identifiant d'un tenant, l'identifiant d'un service producteur ou une requête DSL sur des Units. Il est possible de lancer un audit de l'existence des fichiers uniquement, ou de lancer un audit vérifiant l'existence et l'intégrité des fichiers en même temps.

2.3.1.Processus d'audit d'existence des fichiers (vision métier)

Pour chaque objet du tenant choisi ou chaque objet appartenant au service producteur, l'audit va vérifier :

- Que la stratégie de stockage définie dans le groupe d'objet existe bien sur la plateforme
- Que tous les fichiers correspondant aux objets existent sur les offres déclarées dans la stratégie, dans un nombre de copie spécifié via la stratégie de stockage

2.3.2.Processus d'audit d'intégrité des fichiers (vision métier)

Si l'audit d'intégrité des objets est lancé, le processus va également vérifier que les empreintes des objets stockés en base de données sont bien les mêmes que les empreintes fournies par les espaces de stockage, alors recalculées à la demande de l'audit.

Dans une première étape technique, le processus prépare la liste des groupes d'objets à auditer afin de paralléliser la tâche. Dans un second temps, il effectue la vérification elle-même. A la fin du processus d'audit, un rapport est généré.

2.3.3.Processus de préparation de l'audit (STP_PREPARE_AUDIT)

2.3.3.1. Vérification des seuils de limitation de traitement des unités archivistiques CHECK_DISTRIBUTION_THRESHOLD

- **Règle** : tâche consistant à vérifier les seuils de limitation de traitement des unités archivistiques
- **Type** : bloquant
- **Statuts** :
  - OK : la vérification des seuils de limitation de traitement des unités archivistiques a bien été effectuée (CHECK_DISTRIBUTION_THRESHOLD.OK = Succès de la vérification des seuils de limitation de traitement des unités archivistiques)
  - KO : une incohérence à été détectée entre le seuil et le nombre d'unités archivistiques à traiter. (CHECK_DISTRIBUTION_THRESHOLD.KO = Échec de la vérification des seuils de limitation de traitement des unités archivistiques)
  - FATAL : une erreur technique est survenue lors de la vérification des seuils de limitation de traitement des unités archivistiques (CHECK_DISTRIBUTION_THRESHOLD.FATAL = Erreur technique lors de la vérification des seuils de limitation de traitement des unités archivistiques)

2.3.3.2. Récupération de la configuration des stratégie de stockage de la plateforme PREPARE_STORAGE_STRATEGIES (PrepareStorageStrategiesPlugin.java)

- **Règle** : tâche consistant à charger la configuration des stratégies de la plateforme
- **Type** : bloquant
- **Statuts** :
  - OK : les données ont été récupérées et enregistrées dans un fichier de travail avec succès (PREPARE_STORAGE_STRATEGIES,OK = Succès de la récupération de la liste des stratégies de stockage)
  - FATAL : une erreur technique est survenue lors de la création de la liste (PREPARE_STORAGE_STRATEGIE.FATAL = Erreur technique lors de la récupération de la liste des stratégies de stockage)

2.3.3.3. Création de la liste des groupes d'objets LIST_OBJECTGROUP_ID (PrepareAuditActionHandler.java)

- **Règle** : tâche consistant à créer la liste des groupes d'objets à auditer
- **Type** : bloquant
- **Statuts** :
  - OK : la liste a été créée avec succès (LIST_OBJECTGROUP_ID.OK = Succès de la création de la liste des groupes d'objets à auditer)
  - FATAL : une erreur technique est survenue lors de la création de la liste (LIST_OBJECTGROUP_ID.FATAL = Erreur technique lors de la création de la liste des groupes d'objets à auditer)

2.3.4.Processus d'exécution de l'audit (STP_AUDIT)

2.3.4.1. Audit de la vérification des objets AUDIT_CHECK_OBJECT (AuditCheckObjectPlugin.java)

- **Règle** : tâche consistant à organiser et lancer l'action d'audit de la vérification des objets.
- **Type** : bloquant
- **Statuts** :
  - OK : l'action d'audit de la vérification des objets s'est terminée en OK (AUDIT_CHECK_OBJECT.OK = Succès de l'audit de la vérification des objets)
  - KO : l'action d'audit de la vérification des objets s'est terminée en KO (AUDIT_CHECK_OBJECT.KO = Échec de l'audit de la vérification des objets : au moins un objet demandé n'existe pas ou des stratégies de stockage n'existent pas sur la plateforme)
  - FATAL : une erreur technique est survenue lors du lancement de l'action d'audit (AUDIT_CHECK_OBJECT.FATAL = Erreur technique lors de l'audit de la vérification des objets)

La tâche Audit_Check_Object contient le traitement suivant :

2.3.4.2. Audit de l'existence et de l'intégrité des objets AUDIT_CHECK_OBJECT.AUDIT_CHECK_OBJECT

- **Règle** : traitement consistant à auditer l'existence et l'intégrité des objets
  - La stratégie de stockage du groupe d'objets est conforme à celle du moteur de stockage
  - Les fichiers correspondant aux objets, déclarés dans le groupe d'objets, existent bien sous le même nom dans les offres de stockage
- **Type** : bloquant

- **Statuts :**
  - OK : tous les objets de tous les groupes d'objet audités existent bien sur les offres de stockage et leurs empreintes sont identiques entre celles enregistrées en base de données et celles recalculées par les offres de stockage (AUDIT_CHECK_OBJECT.AUDIT_CHECK_OBJECT.OK = Succès de l'audit de l'existence et de l'intégrité des objets)
  - KO : au moins un objet n'est pas intègre pour les groupes d'objets, audités (AUDIT_CHECK_OBJECT.AUDIT_CHECK_OBJECT.KO = Échec de l'audit de l'existence de fichiers)
  - Warning : il n'y a aucun objet à auditer (cas par exemple d'un producteur sans objets) (AUDIT_CHECK_OBJECT.AUDIT_CHECK_OBJECT.WARNING = Avertissement lors de l'audit de l'existence et de l'intégrité des objets : au moins un élément n'a pas d'objet binaire à vérifier)
  - FATAL : une erreur technique est survenue lors de l'audit de l'existence et de l'intégrité des objets (AUDIT_CHECK_OBJECT.AUDIT_FILE_EXISTING.FATAL = Erreur technique lors de l'audit de l'existence et de l'intégrité des objets)

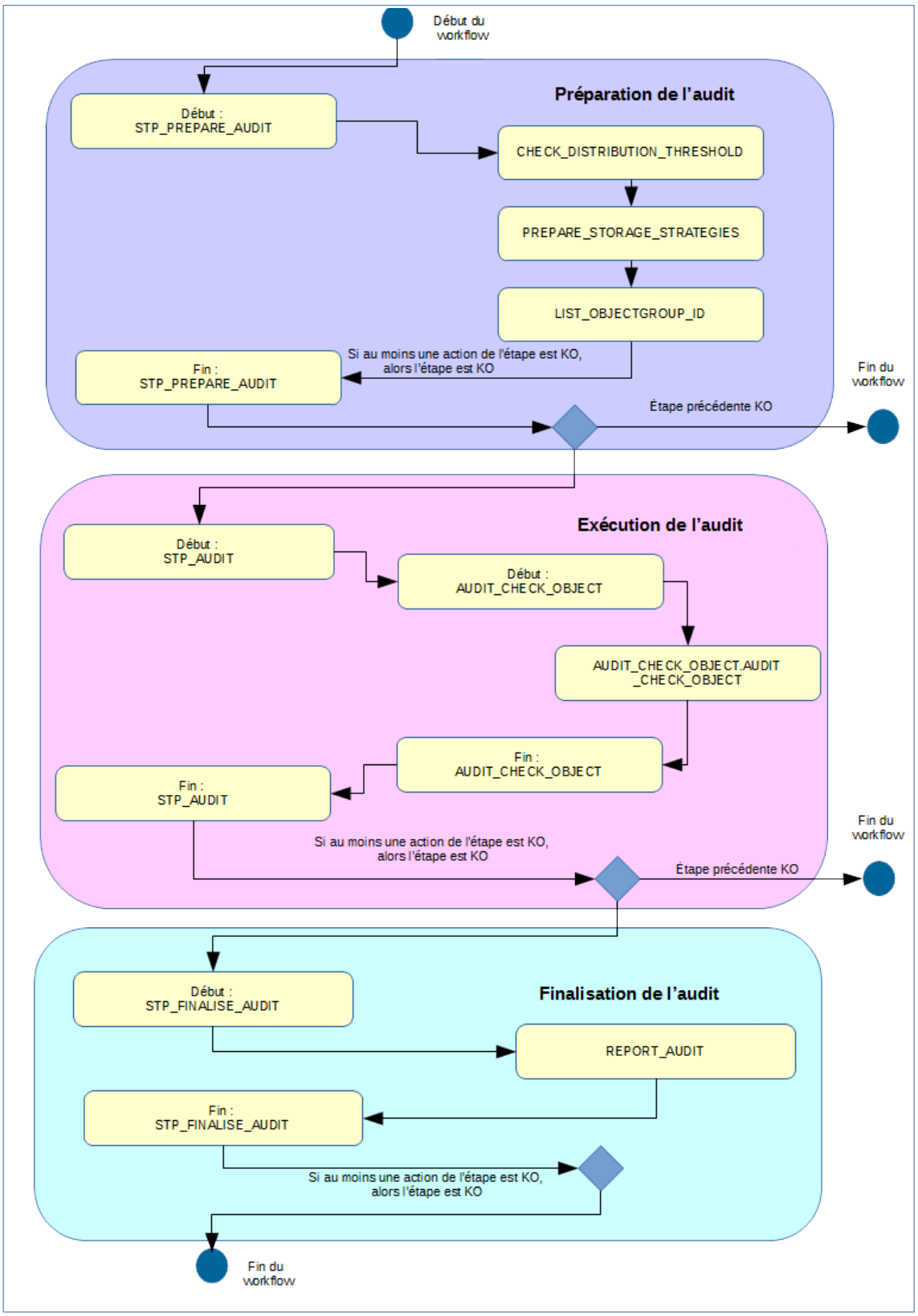
2.3.5.Processus de finalisation de l'audit (STP_FINALISE_AUDIT)

Notification de la fin d'audit REPORT_AUDIT (GenerateAuditReportActionHandler.java)

- **Règle** : tâche consistant à générer le rapport d'audit
- **Type** : bloquant
- **Statuts :**
  - OK : le rapport a été créé avec succès (REPORT_AUDIT.OK = Succès de la notification de la fin de l'audit)
  - FATAL : une erreur technique est survenue lors de la création du rapport d'audit (REPORT_AUDIT.OK.FATAL = Erreur technique lors de la notification de la fin de l'audit)

2.3.6.Structure du workflow de l'audit de l'existence et de l'intégrité des fichiers

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



2.3.7.Rapport d’audit

Le rapport d’audit est un fichier JSON généré par la solution logicielle Vitam lorsqu’une opération d’audit se termine. Cette section décrit la manière dont ce rapport est structuré.

2.3.7.1. Exemple de JSON : rapport d’audit d’intégrité et d’existence KO

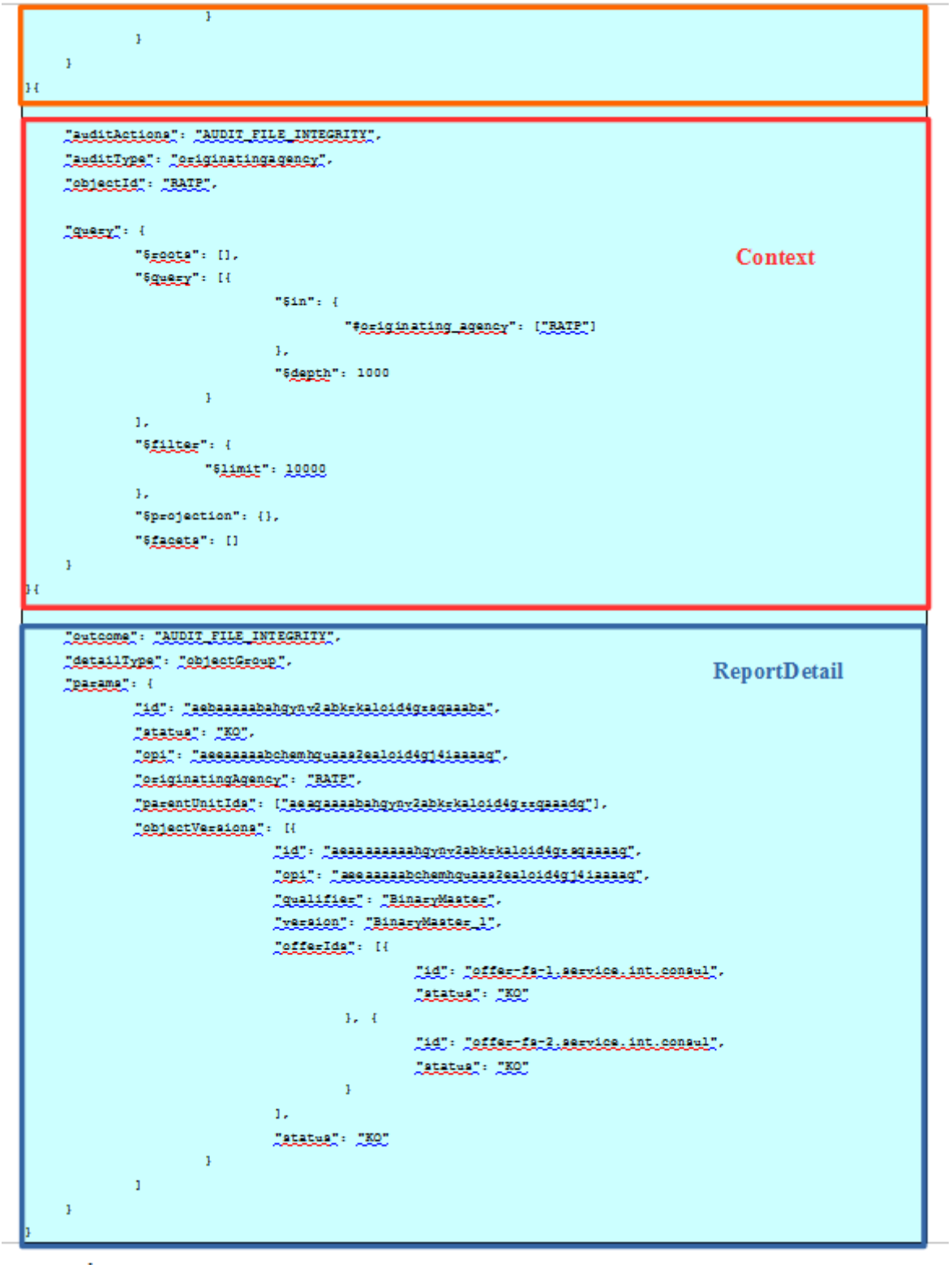
2.3.7.2.

OperationSummary


```
{
    "tenant": 8,
    "evId": "aeeaaaaabchemhquaajkyaloieeby3qaaaaq",
    "evType": "PROCESS_AUDIT",
    "outcome": "KO",
    "outDetail": "AUDIT_CHECK_OBJECT.AUDIT_CHECK_OBJECT.KO",
    "outMsg": "Échec de l'audit de l'existence et de l'intégrité des objets Detail=  OK:3 KO:1",
    "rightsStatementIdentifier": {
            "AccessContract": "AC-000006"
    },
    "evDetData": null
}{
```


ReportSummary


```
    "evStartDateTime": "2019-11-06T14:04:41.360",
    "evEndDateTime": "2019-11-06T14:04:44.841",
    "reportType": "AUDIT",
    "vitamResults": {
            "OK": 3,
            "KO": 1,
            "WARNING": 0,
            "total": 4
    },
    "extendedInfo": {
            "nbObjectGroups": 4,
            "nbObjects": 4,
            "opis": ["aeeaaaaabchemhquaas2ealoid4gj4iaaaaq"],
            "globalResults": {
                    "objectGroupsCount": {
                            "OK": 3,
                            "WARNING": 0,
                            "KO": 1
                    },
                    "objectsCount": {
                            "OK": 3,
                            "WARNING": 0,
                            "KO": 1
                    }
            },
            "originatingAgencyResults": {
                    "RATP": {
                            "objectGroupsCount": {
                                    "OK": 3,
                                    "WARNING": 0,
                                    "KO": 1
                            },
                            "objectsCount": {
                                    "OK": 3,
                                    "WARNING": 0,
                                    "KO": 1
                            }
```



2.3.7.3. Détail du rapport

La partie « OperationSummary », c'est-à-dire le bloc à la racine du rapport et correspondant au résumé de l'opération (sans indentation) est composée des champs suivants :

- « tenant » : tenant sur lequel l'opération d'audit a été lancée
- « evId » : identifiant de l'événement

- « evType » : code du type de l'opération
- « outcome »: statut de l'événement
- « outcomeMsg » : détail du résultat de l'événement.
- « rightsStatementIdentifier »:identifiant des données référentielles en vertu desquelles l'opération peut s'exécuter.
- « evDetData » : détails des données l'événement

La partie « ReportSummary », c'est-à-dire le bloc situé sous la racine du rapport et correspondant au résumé du rapport est composée des champs suivants :

- « evStartDateTime » : date du début de l'opération (evDateTime de l'event master de l'opération dans le Journal des Opérations)
- « evEndDateTime » : date de fin d'opération (dernier evDateTime dans l'opération dans le Journal des Opérations)
- « reportType »: corresponds au modèle du rapport. Chaque opération de masse (audit, élimination…) aura un « reportType » associé
- « vitamResults » : est le nombre de OK, KO, WARNING de l'opération ainsi que le total de ces 3 statuts.
- « extendedInfo » : partie libre où chaque type de rapport pourra ajouter des informations qui lui est propre. Ici on y trouve :
  - le nombre total de groupes d'objets et d'objets audités pour l'ensemble de l'opération (« nbObjectGroups », « nbObjects »)
  - les identifiants des opérations d'entrée concernant ces groupes d'objets et objets (« opis »)
  - un résultat global (« globalResults ») remontant le nombre de groupes d'objets et d'objets aux statuts « OK », « KO » et « WARNING » (« objectGroupsCount » et « objectsCount »)
  - Un résultat par service producteur (« originatingAgencyResultats ») remontant le nombre de groupes d'objets et d'objets aux statuts « OK », « KO » et « WARNING » « objectGroupsCount » et « objectsCount »).

La partie « Context » correspond à la requête DSL utilisée pour créer la distribution sur chaque unité archivistique.On y retrouve outre la requête elle-même le type d'audit réalisée (« auditActions », ici AUDIT_FILE_INTEGRITY), l'élément sur lequel l'audit à été lancé. Celui-ci peut-être par « tenant », ou par « originatingagency » (« auditType », ici ORIGINATINGAGENCY) et enfin, identifiant de l'élément (tenant ou service producteur), (« objectId », ici « RATP).

La partie « ReportDetail » contient les détails pour chaque groupe d'objets et objets audités dont le statut est « KO »

La liste des éléments singuliers générant un avertissement « auditWarning » décrit les identifiants des services producteurs ayant généré un avertissement. Dans le cas de l'audit de l'existence des fichiers, une alerte correspond au fait qu'un service producteur n'a aucun objet à auditer. Cette liste est donc l'ensemble des services producteurs concernés par l'audit mais dont il n'existe aucun objet à auditer.

2.4.Workflow d'audit de cohérence des fichiers

Cette section décrit le processus (workflow) d'audit de cohérence des fichiers mis en place dans la solution logicielle Vitam.

Celui-ci est défini dans le fichier *« EvidenceAuditWorkflow.json »* (situé ici : sources/processing/processing-management/src/main/resources/workflows).

Toutes les étapes, tâches et traitements sont journalisées dans le journal des opérations et décrivent le processus (clé et description de la clé associée dans le journal des opérations) tel qu'implémenté dans la version actuelle de la solution logicielle Vitam.

2.4.1.Processus d'audit de cohérence des fichiers (vision métier)

Le processus d'audit de cohérence permet de vérifier la cohérence entre les signatures calculées pour chaque élément audité, en comparant celle présente dans le journal sécurisé, avec celle présente dans la base de donnée, et celle de l'offre de stockage.

L'audit s'applique au niveau des unités archivistiques, des objets et des groupes d'objets.

2.4.2.Processus de préparation d'audit (STP_EVIDENCE_AUDIT_PREPARE)

2.4.2.1. Vérification des seuils de limitation de traitement des unités archivistiques CHECK_DISTRIBUTION_THRESHOLD

- **Règle** : tâche consistant à vérifier les seuils de limitation de traitement des unités archivistiques
- **Type** : bloquant
- **Statuts** :
  - OK : la vérification des seuils de limitation de traitement des unités archivistiques a bien été effectuée (CHECK_DISTRIBUTION_THRESHOLD.OK = Succès de la vérification des seuils de limitation de traitement des unités archivistiques)
  - KO : une incohérence à été détectée entre le seuil et le nombre d'unités archivistiques à traiter. (CHECK_DISTRIBUTION_THRESHOLD.KO = Échec de la vérification des seuils de limitation de traitement des unités archivistiques)
  - FATAL : une erreur technique est survenue lors de la vérification des seuils de limitation de traitement des unités archivistiques (CHECK_DISTRIBUTION_THRESHOLD.FATAL = Erreur technique lors de la vérification des seuils de limitation de traitement des unités archivistiques)

2.4.2.2. Récupération de la configuration des stratégie de stockage de la plateforme PREPARE_STORAGE_STRATEGIES (PrepareStorageStrategiesPlugin.java)

- **Règle** : tâche consistant à charger la configuration des stratégies de la plateforme
- **Type** : bloquant
- **Statuts** :
  - OK : les données ont été récupérées et enregistrées dans un fichier de travail avec succès (PREPARE_STORAGE_STRATEGIES,OK = Succès de la récupération de la liste des stratégies de stockage)
  - FATAL : une erreur technique est survenue lors de la création de la liste (PREPARE_STORAGE_STRATEGIE.FATAL = Erreur technique lors de la récupération de la liste des stratégies de stockage)

2.4.2.3. Création de la liste à auditer EVIDENCE_AUDIT_LIST_OBJECT (EvidenceAuditPrepare.java)

- **Règle** : tâche consistant à établir la liste des objets à auditer
- **Type** : bloquant
- **Statuts :**
  - OK : la liste a été créée avec succès (EVIDENCE_AUDIT_LIST_OBJECT.OK = Succès de la création de la liste à auditer)
  - KO : échec de la création de la liste à auditer (EVIDENCE_AUDIT_LIST_OBJECT.KO = Échec de la création de la liste à auditer)
  - FATAL : une erreur technique est survenue lors de la création de la liste (EVIDENCE_AUDIT_LIST_OBJECT.FATAL = Erreur technique lors de la création de la liste à auditer)

2.4.3. Processus de récupération des données de la base (STP_EVIDENCE_AUDIT_CHECK_DATABASE)

2.4.3.1. Récupération des données dans la base de données EVIDENCE_AUDIT_CHECK_DATABASE (EvidenceAuditDatabaseCheck.java)

- **Règle** : tâche consistant à récupérer les informations nécessaires à l'audit dans la base de données
- **Type** : bloquant
- **Statuts** :
  - OK : la récupération des données dans la base de données est un succès (EVIDENCE_AUDIT_CHECK_DATABASE.OK = Succès de la récupération des données dans la base de données)
  - KO : la récupération des données dans la base de donnée est un échec (EVIDENCE_AUDIT_CHECK_DATABASE.KO = Échec de la récupération des données dans la base de données)
  - FATAL : une erreur technique est survenue dans la récupération des données dans la base de données (EVIDENCE_AUDIT_CHECK_DATABASE.FATAL = Erreur technique lors de la récupération des données dans la base de données)
  - WARNING : avertissement lors de la récupération des données dans la base de données (EVIDENCE_AUDIT_CHECK_DATABASE.WARNING = Avertissement lors de la récupération des données dans la base de données)

2.4.4. Processus de préparation des signatures à partir des fichiers sécurisés (STP_EVIDENCE_AUDIT_LIST_SECURED_FILES_TO_DOWNLOAD)

2.4.4.1. Préparation de la liste des signatures dans les fichiers sécurisés EVIDENCE_AUDIT_LIST_SECURED_FILES_TO_DOWNLOAD (EvidenceAuditListSecuredFiles.java)

- **Règle** : tâche consistant à préparer la liste des signatures des objets, groupes d'objets ou unités archivistiques archivées, dans les fichiers sécurisés
- **Type** : bloquant
- **Statuts** :
  - OK : la préparation de la liste des signatures dans les fichiers sécurisés est un succès (EVIDENCE_AUDIT_LIST_SECURED_FILES_TO_DOWNLOAD.OK = Succès de la préparation

de la liste des signatures dans les fichiers sécurisés)

- KO : la préparation de la liste des signatures dans les fichiers sécurisés est un échec (EVIDENCE_AUDIT_LIST_SECURED_FILES_TO_DOWNLOAD.KO = Échec de la préparation de la liste des signatures dans les fichiers sécurisés)
- WARNING : avertissement lors de la préparation de la liste des signatures (EVIDENCE_AUDIT_LIST_SECURED_FILES_TO_DOWNLOAD.WARNING = Avertissement lors de la préparation de la liste des signatures dans les fichiers sécurisés)
- FATAL : une erreur technique est survenue lors de la préparation de la liste des signatures dans les fichiers sécurisés (EVIDENCE_AUDIT_LIST_SECURED_FILES_TO_DOWNLOAD.FATAL = Erreur technique lors de la préparation de la liste des signatures dans les fichiers sécurisés)

2.4.5. Processus d'extraction des signatures à partir des fichiers sécurisés (STP_EVIDENCE_AUDIT_EXTRACT_ZIP_FILE)

2.4.5.1. Extraction des signatures à partir des fichiers sécurisés EVIDENCE_AUDIT_EXTRACT_ZIP_FILE (EvidenceAuditExtractFromZip.java)

- **Règle** : tâche consistant à extraire les signatures des objets, groupes d'objets ou unités archivistiques archivées, dans les fichiers sécurisés
- **Type** : bloquant
- **Statuts** :
  - OK : l'extraction des signatures à partir des fichiers sécurisés est un succès (EVIDENCE_AUDIT_EXTRACT_ZIP_FILE.OK = Succès de l'extraction des signatures à partir des fichiers sécurisés)
  - KO : l'extraction des signatures à partir des fichiers sécurisés est un échec (EVIDENCE_AUDIT_EXTRACT_ZIP_FILE.KO = Échec de l'extraction des signatures à partir des fichiers sécurisés)
  - WARNING : avertissement lors de l'extraction des signatures à partir des fichiers sécurisés (STP_EVIDENCE_AUDIT_EXTRACT_ZIP_FILE.WARNING = Avertissement lors de l'extraction des signatures à partir des fichiers sécurisés)
  - FATAL : une erreur technique est survenue lors de l'extraction des signatures à partir des fichiers sécurisés (EVIDENCE_AUDIT_EXTRACT_ZIP_FILE.FATAL = Erreur technique lors de l'extraction des signatures à partir des fichiers sécurisés)

2.4.6. Processus de préparation des rapports pour chaque objet, groupe d'objets ou unité audité (STP_EVIDENCE_AUDIT_PREPARE_GENERATE_REPORTS)

2.4.6.1. Création du rapport pour chaque unité archivistique ou objet ou groupe d'objets EVIDENCE_AUDIT_PREPARE_GENERATE_REPORTS (EvidenceAuditGenerateReports.java)

- **Règle** : tâche consistant à créer le rapport pour chaque unité archivistique, objet ou groupe d'objets audité
- **Type** : bloquant
- **Statuts :**
  - OK : La création du rapport pour chaque unité archivistique ou objet ou groupe d'objets est un succès (EVIDENCE_AUDIT_PREPARE_GENERATE_REPORTS.OK = Succès de la création du rapport pour chaque unité archivistique ou objet ou groupe d'objets)
  - KO : la création du rapport pour chaque unité archivistique ou objet ou groupe d'objets est un échec (EVIDENCE_AUDIT_PREPARE_GENERATE_REPORTS.KO = Échec de la création

du rapport pour chaque unité archivistique ou objet ou groupe d'objets)

- FATAL : une erreur technique est survenue de la création du rapport pour chaque unité archivistique ou objet ou groupe d'objets (EVIDENCE_AUDIT_PREPARE_GENERATE_REPORTS.FATAL = Erreur technique lors de la création du rapport pour chaque unité archivistique ou objet ou groupe d'objets)
- WARNING : avertissement lors de la création du rapport pour chaque unité archivistique ou objet ou groupe d'objets (EVIDENCE_AUDIT_PREPARE_GENERATE_REPORTS.WARNING = Avertissement lors de la création du rapport pour chaque unité archivistique ou objet ou groupe d'objets)

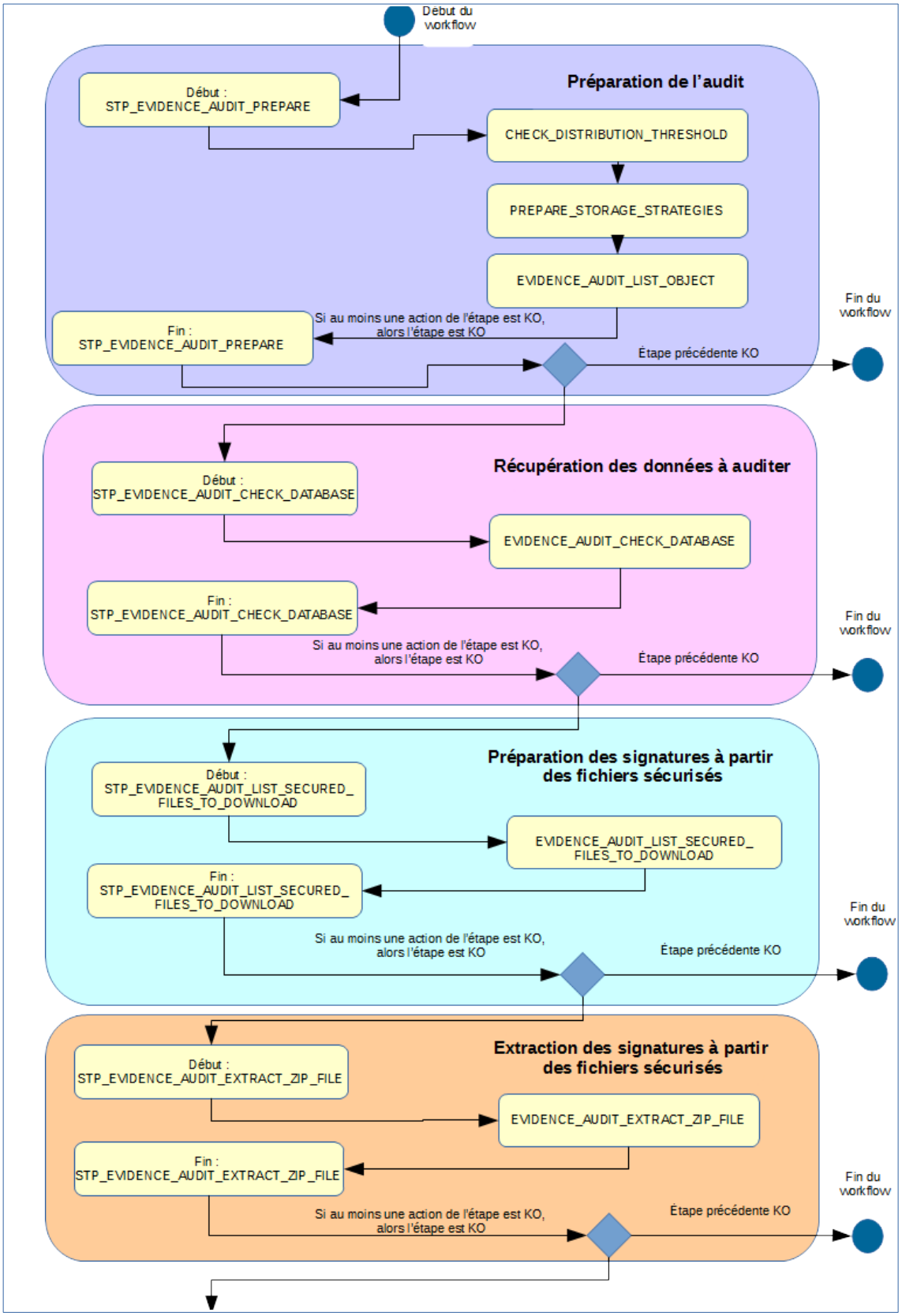
2.4.7. Processus de finalisation d'audit et génération du rapport final (STP_EVIDENCE_AUDIT_FINALIZE)

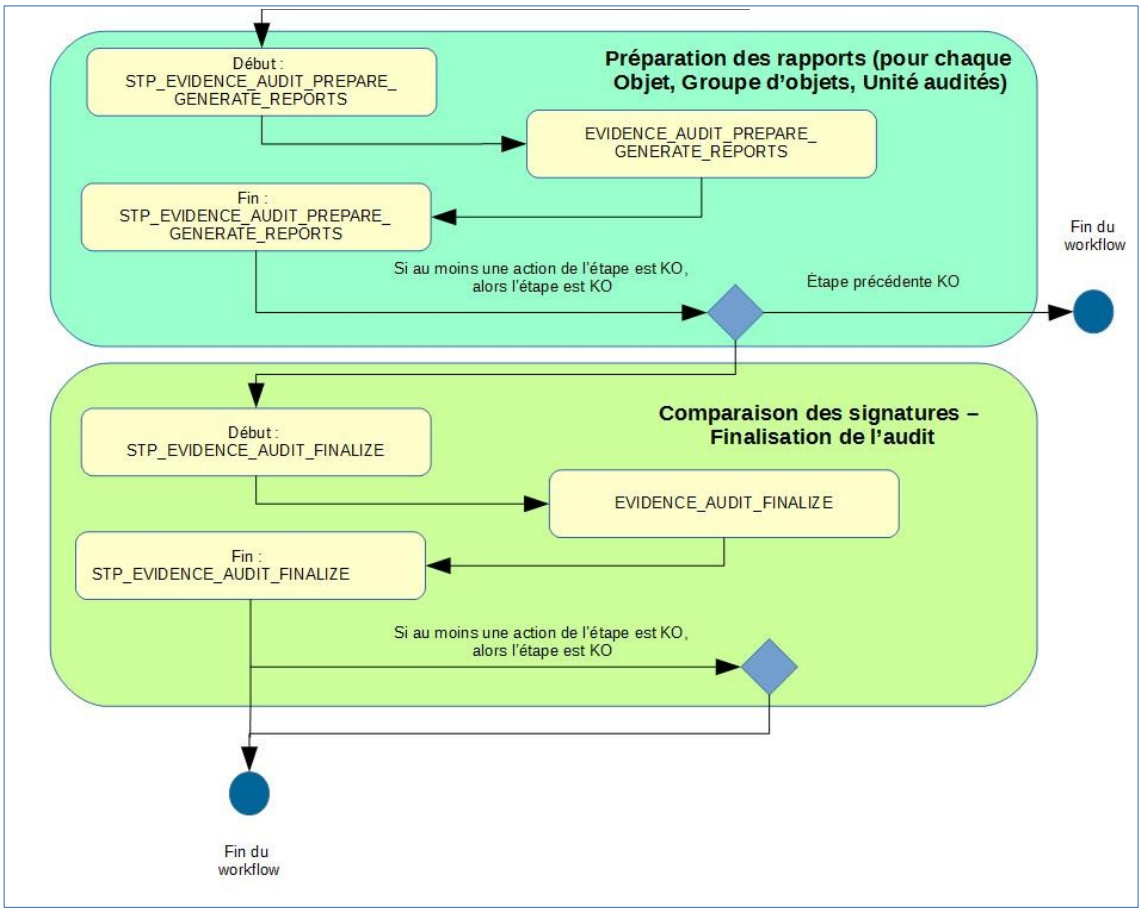
2.4.7.1. Création du rapport d'audit de cohérence EVIDENCE_AUDIT_FINALIZE (EvidenceAuditFinalize.java)

- **Règle** : tâche consistant à créer le rapport permettant de comparer les signatures extraites des fichiers sécurisés avec les données de la base de données et de l'offre de stockage
- **Type** : bloquant
- **Statuts :**
  - OK : la création du rapport d'audit de cohérence est un succès (EVIDENCE_AUDIT_FINALIZE.OK = Succès de la création du rapport d'audit de cohérence)
  - KO : la création du rapport d'audit de cohérence est un échec (EVIDENCE_AUDIT_FINALIZE.KO = Échec de la création du rapport d'audit de cohérence)
  - FATAL : une erreur technique est survenue lors de la création du rapport d'audit de cohérence) (EVIDENCE_AUDIT_FINALIZE.FATAL = Erreur technique lors de la création du rapport d'audit de cohérence)

2.4.8.Structure du workflow d’audit de cohérence des fichiers

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


Début du workflow
Préparation de l'audit
Début : STP_EVIDENCE_AUDIT_PREPARE
CHECK_DISTRIBUTION_THRESHOLD
PREPARE_STORAGE_STRATEGIES
EVIDENCE_AUDIT_LIST_OBJECT
Fin : STP_EVIDENCE_AUDIT_PREPARE
Si au moins une action de l'étape est KO, alors l'étape est KO
Étape précédente KO
Fin du workflow
Récupération des données à auditer
Début : STP_EVIDENCE_AUDIT_CHECK_DATABASE
EVIDENCE_AUDIT_CHECK_DATABASE
Fin : STP_EVIDENCE_AUDIT_CHECK_DATABASE
Si au moins une action de l'étape est KO, alors l'étape est KO
Étape précédente KO
Fin du workflow
Préparation des signatures à partir des fichiers sécurisés
Début : STP_EVIDENCE_AUDIT_LIST_SECURED_FILES_TO_DOWNLOAD
EVIDENCE_AUDIT_LIST_SECURED_FILES_TO_DOWNLOAD
Fin : STP_EVIDENCE_AUDIT_LIST_SECURED_FILES_TO_DOWNLOAD
Si au moins une action de l'étape est KO, alors l'étape est KO
Étape précédente KO
Fin du workflow
Extraction des signatures à partir des fichiers sécurisés
Début : STP_EVIDENCE_AUDIT_EXTRACT_ZIP_FILE
EVIDENCE_AUDIT_EXTRACT_ZIP_FILE
Fin : STP_EVIDENCE_AUDIT_EXTRACT_ZIP_FILE
Si au moins une action de l'étape est KO, alors l'étape est KO
Étape précédente KO
Fin du workflow


Préparation des rapports (pour chaque Objet, Groupe d'objets, Unité audités)
Début : STP_EVIDENCE_AUDIT_PREPARE_ GENERATE_REPORTS
EVIDENCE_AUDIT_PREPARE_ GENERATE_REPORTS
Fin : STP_EVIDENCE_AUDIT_PREPARE_ GENERATE_REPORTS
Si au moins une action de l'étape est KO, alors l'étape est KO
Étape précédente KO
Fin du workflow
Comparaison des signatures – Finalisation de l'audit
Début : STP_EVIDENCE_AUDIT_FINALIZE
EVIDENCE_AUDIT_FINALIZE
Fin : STP_EVIDENCE_AUDIT_FINALIZE
Si au moins une action de l'étape est KO, alors l'étape est KO
Fin du workflow

2.4.9.Rapport d'audit de cohérence

Le rapport d'audit de cohérence est un fichier JSONL généré par la solution logicielle Vitam lorsqu'une opération d'audit se termine. En cas de succès (EVIDENCE_AUDIT.OK), le rapport ne précise pas les unités archivistiques, les groupes d'objets techniques et les objets techniques qui ont fait l'objet de l'audit. Dans les cas de KO et de WARNING le rapport retourne les informations détaillées ci-dessous.

2.4.9.1. Exemple de JSON : rapport d'audit de cohérence d'un objet technique KO

OperationSummary


```
{
"tenant": 0,
    "evId": "aeeaaaaachemhqaadhalqjjvntqaaaaq",
    "evType": "EVIDENCE_AUDIT",
    "outcome": "OK",
    "outDetail": "STP_EVIDENCE_AUDIT_PREPARE_GENERATE_REPORTS.OK",
    "outMsg": "Succ\u00E8s du processus de cr\u00E9ation du rapport pour chaque unit\u00E9 archivistique ou objet ou
groupe d'objets",
    "rightsStatementIdentifier": {
        "AccessContract": "ContratTNR"
    },
    "evDetData": {}
```


ReportSummary


```
"evStartDateTime":"2019-11-08T09:55:40.113",
    "evEndDateTime":"2019-11-08T09:55:45.913",
    "reportType":"EVIDENCE_AUDIT",
    "vitamResults": {
        "OK": 2,
        "KO": 1,
        "WARNING": 0,
        "total": 3
    },
    "extendedInfo": {
        "nbObjectGroups": 1,
        "nbArchiveUnits": 1,
        "nbObjects": 1,
        "globalResults": {
            "objectGroupsCount": {
                "OK": 1,
                "WARNING": 0,
                "KO": 0
            },
            "archiveUnitsCount": {
                "OK": 1,
                "WARNING": 0,
                "KO": 0
            },
            "objectsCount": {
                "OK": 0,
                "WARNING": 0,
```

2.4.9.2. Détail du rapport

La partie « OperationSummary », c'est-à-dire le bloc à la racine du rapport et correspondant au résumé de l'opération (sans indentation) est composée des champs suivants :

- « tenant » : tenant sur lequel l'opération d'audit a été lancée
- « evId » : identifiant de l'événement
- « evType » : code du type de l'opération
- « outcome »: statut de l'événement
- « outcomeMsg » : détail du résultat de l'événement.
- « rightsStatementIdentifier »:identifiant des données référentielles en vertu desquelles l'opération peut s'exécuter.
- « evDetData » : détails des données l'événement

La partie « ReportSummary », c'est-à-dire le bloc situé sous la racine du rapport et correspondant au résumé du rapport est composée des champs suivants :

- « evStartDateTime » : date du début de l'opération (evDateTime de l'event master de l'opération dans le journal des opérations) ;
- « evEndDateTime » : date de fin d'opération (dernier evDateTime dans l'opération dans le journal des opérations) ;
- « reportType »: corresponds au modèle du rapport. Chaque opération de masse (audit, élimination…) a un « reportType » associé. Dans le cas d'un audit de cohérence, la valeur du champ est égale à « EVIDENCE_AUDIT » ;
- « vitamResults » : est le nombre de OK, KO, WARNING de l'opération ainsi que le total de ces 3 statuts ;
- « extendedInfo » : partie libre où chaque type de rapport contient des informations qui lui est

propre. Ici on y trouve :

- le nombre total d'unités archivistiques auditées pour l'ensemble de l'opération (« nbArchiveUnits »),
- le nombre total de groupes d'objets audités pour l'ensemble de l'opération (« nbObjectGroups »),
- le nombre total d'objets audités pour l'ensemble de l'opération (« nbObjects »),
- un résultat global (« globalResults ») remontant le nombre d'unités archivistiques, de groupes d'objets et d'objets aux statuts « OK », « KO » et « WARNING » (« objectGroupsCount », « archiveUnitsCount » et « objectsCount »)

La partie « Context » correspond à la requête DSL utilisée pour créer la distribution sur chaque unité archivistique. Dans le rapport d'un audit de cohérence, on y retrouve uniquement la requête.

La partie « ReportDetail » contient les détails de l'opération d'audit de cohérence effectuée sur chacune des unités archivistiques, des groupes d'objets techniques et/ou des objets techniques, uniquement quand ces derniers sont aux statuts « KO » ou « WARNING ».

- « identifier » : identifiant de l'objet, groupe d'objets ou unité archivistique audité ;
- « status » : résultat de l'opération pour le groupe d'objets, l'objet ou l'unité archivistique concerné, pouvant correspondre aux valeurs suivantes : « KO » ou « WARNING » ;
- « objectType » : type de l'objet audité, pouvant correspondre aux valeurs suivantes : « UNIT », « OBJECTGROUP » ou « OBJECT » ;
- « message » : message qui signale une incohérence entre les signatures des fichiers sécurisés, des offres de stockage et de la base de données ;
- « securedHash » : empreindre du journal sécurisé de l'unité archivistique, objet ou groupe d'objets ;
- « offersHashes » : signatures de l'élément audité (unité archivistique ou groupes d'objets techniques) dans les différentes offres de stockage ;
- « strategyId » : identifiant de la stratégie de stockage.

2.5.Workflow de l'audit correctif

Cette section décrit le processus (workflow) de l'audit correctif des fichiers mis en place dans la solution logicielle Vitam. Cette opération peut être effectuée suite à l'audit de cohérence. Cette action est effectuée via API. Cependant, dans un souci de démonstration et à titre d'écran expérimental un lancement de l'audit correctif peut-être effectué à partir de l'IHM recette (cf 2.4 Test Audit correctif).

Toutes les étapes, tâches et traitements sont journalisées dans le journal des opérations et décrivent le processus (clé et description de la clé associée dans le journal des opérations) tel qu'implémenté dans la version actuelle de la solution logicielle Vitam.

2.5.1.Processus d'audit correctif des fichiers (vision métier)

Cette action a pour but de corriger des objets défaillants. Cette correction fait suite à l'audit de cohérence. L'audit correctif veut résoudre une situation anormale : l'empreinte des fichiers ne correspond plus à l'empreinte enregistrée en base et sur les espaces de stockage. Afin de corriger et de rectifier ces erreurs, l'audit correctif va supprimer la copie défaillante (si cela est possible) et restaurer à partir de la stratégie de stockage une copie saine (copie dont l'empreinte est OK). Les premières parties du workflow de correction de l'audit suivent les mêmes étapes que l'audit de cohérence, le traitement de correction et de récupération des données dans les offres de stockage ou la base de données intervient dans la seconde partie du workflow.

2.5.2.Processus de préparation d'audit (STP_EVIDENCE_AUDIT_PREPARE)

2.5.2.1. Vérification des seuils de limitation de traitement des unités archivistiques CHECK_DISTRIBUTION_THRESHOLD

- **Règle** : étape consistant à vérifier les seuils de limitation de traitement des unités archivistiques
- **Type** : bloquant
- **Statuts** :
  - OK : la vérification des seuils de limitation de traitement des unités archivistiques a bien été effectuée (CHECK_DISTRIBUTION_THRESHOLD.OK = Succès de la vérification des seuils de limitation de traitement des unités archivistiques)
  - KO : une incohérence à été détectée entre le seuil et le nombre d'unités archivistiques à traiter (CHECK_DISTRIBUTION_THRESHOLD.KO = Échec de la vérification des seuils de limitation de traitement des unités archivistiques)
  - FATAL : une erreur technique est survenue lors de la vérification des seuils de limitation de traitement des unités archivistiques (CHECK_DISTRIBUTION_THRESHOLD.FATAL = Erreur technique lors de la vérification des seuils de limitation de traitement des unités archivistiques)

2.5.2.2. Récupération de la configuration des stratégie de stockage de la plateforme PREPARE_STORAGE_STRATEGIES (PrepareStorageStrategiesPlugin.java)

- **Règle** : tâche consistant à charger la configuration des stratégies de la plateforme
- **Type** : bloquant
- **Statuts** :
  - OK : les données ont été récupérées et enregistrées dans un fichier de travail avec succès (PREPARE_STORAGE_STRATEGIES.OK = Succès de la récupération de la liste des stratégies de stockage)
  - FATAL : une erreur technique est survenue lors de la création de la liste (PREPARE_STORAGE_STRATEGIES.FATAL = Erreur technique lors de la récupération de la liste des stratégies de stockage)

2.5.2.3. Création de la liste à auditer EVIDENCE_AUDIT_LIST_OBJECT (EvidenceAuditPrepare.java)

- **Règle** : tâche consistant à établir la liste des objets à auditer
- **Type** : bloquant
- **Statuts :**
  - OK : la liste a été créée avec succès (EVIDENCE_AUDIT_LIST_OBJECT.OK = Création de la liste à auditer)
  - KO : échec de la création de la liste à auditer (EVIDENCE_AUDIT_LIST_OBJECT.KO = Échec lors de la création de la liste à auditer)
  - FATAL : une erreur technique est survenue lors de la création de la liste (EVIDENCE_AUDIT_LIST_OBJECT.FATAL = Erreur technique lors de la création de la liste à auditer)

2.5.3. Processus de récupération des données de la base (STP_EVIDENCE_AUDIT_CHECK_DATABASE)

2.5.3.1. Récupération des données dans la base de donnée EVIDENCE_AUDIT_CHECK_DATABASE (EvidenceAuditDatabaseCheck.java)

- **Règle** : tâche consistant à récupérer les informations nécessaires à l'audit dans la base de données
- **Type** : bloquant
- **Statuts** :
  - OK : la récupération des données dans la base de données est un succès (EVIDENCE_AUDIT_CHECK_DATABASE.OK = Succès de la récupération des données dans la base de données)
  - KO : la récupération des données dans la base de données est un échec (EVIDENCE_AUDIT_CHECK_DATABASE.KO = Échec de la récupération des données dans la base de données)
  - FATAL : une erreur technique est survenue dans la récupération des données dans la base de données (EVIDENCE_AUDIT_CHECK_DATABASE.FATAL = Erreur technique lors de la récupération des données dans la base de données)
  - WARNING : avertissement lors de la récupération des données dans la base de données (EVIDENCE_AUDIT_CHECK_DATABASE.WARNING = Avertissement lors de la récupération des données dans la base de données)

2.5.4. Processus de préparation des signatures à partir des fichiers sécurisés (STP_EVIDENCE_AUDIT_LIST_SECURED_FILES_TO_DOWNLOAD)

2.5.4.1. Préparation de la liste des signatures dans les fichiers sécurisés EVIDENCE_AUDIT_LIST_SECURED_FILES_TO_DOWNLOAD (EvidenceAuditListSecuredFiles.java)

- **Règle** : tâche consistant à préparer la liste des signatures des objets, groupes d'objets ou unités archivistiques archivées, dans les fichiers sécurisés
- **Type** : bloquant
- **Statuts** :
  - OK : la préparation de la liste des signatures dans les fichiers sécurisés est un succès (EVIDENCE_AUDIT_LIST_SECURED_FILES_TO_DOWNLOAD.OK = Succès de la préparation

de la liste des signatures dans les fichiers sécurisés)

- KO : la préparation de la liste des signatures dans les fichiers sécurisés est un échec (EVIDENCE_AUDIT_LIST_SECURED_FILES_TO_DOWNLOAD.KO = Échec de la préparation de la liste des signatures dans les fichiers sécurisés)
- WARNING : avertissement lors de la préparation de la liste des signatures (EVIDENCE_AUDIT_LIST_SECURED_FILES_TO_DOWNLOAD.WARNING = Avertissement lors de la préparation de la liste des signatures dans les fichiers sécurisés)
- FATAL : une erreur technique est survenue lors de la préparation de la liste des signatures dans les fichiers sécurisés (EVIDENCE_AUDIT_LIST_SECURED_FILES_TO_DOWNLOAD.FATAL = Erreur technique lors de la préparation de la liste des signatures dans les fichiers sécurisés)

2.5.5.Processus d'extraction des signatures à partir des fichiers sécurisés (STP_EVIDENCE_AUDIT_EXTRACT_ZIP_FILE)

2.5.5.1. Extraction des signatures à partir des fichiers sécurisés EVIDENCE_AUDIT_EXTRACT_ZIP_FILE (EvidenceAuditExtractFromZip.java)

- **Règle** : tâche consistant à extraire les signatures des objets, groupes d'objets ou unités archivistiques archivées, dans les fichiers sécurisés
- **Type** : bloquant
- **Statuts** :
  - OK : l'extraction des signatures à partir des fichiers sécurisés est un succès (EVIDENCE_AUDIT_EXTRACT_ZIP_FILE.OK = Succès de l'extraction des signatures à partir des fichiers sécurisés)
  - KO : l'extraction des signatures à partir des fichiers sécurisés est un échec (EVIDENCE_AUDIT_EXTRACT_ZIP_FILE.KO = Échec de l'extraction des signatures à partir des fichiers sécurisés)
  - WARNING : avertissement lors de l'extraction des signatures à partir des fichiers sécurisés (STP_EVIDENCE_AUDIT_EXTRACT_ZIP_FILE.WARNING = Avertissement lors de l'extraction des signatures à partir des fichiers sécurisés)
  - FATAL : Une erreur technique est survenue lors de l'extraction des signatures à partir des fichiers sécurisés (EVIDENCE_AUDIT_EXTRACT_ZIP_FILE.FATAL = Erreur technique lors de l'extraction des signatures à partir des fichiers sécurisés)

2.5.6.Processus de préparation des rapports pour chaque objet, groupe d'objets ou unité audité (STP_EVIDENCE_AUDIT_PREPARE_GENERATE_REPORTS)

2.5.6.1. Création du rapport pour chaque unité archivistique ou objet ou groupe d'objets EVIDENCE_AUDIT_PREPARE_GENERATE_REPORTS (EvidenceAuditGenerateReports.java)

- **Règle** : tâche consistant à créer le rapport pour chaque unité archivistique, objet ou groupe d'objets audité
- **Type** : bloquant
- **Statuts :**
  - OK : la création du rapport pour chaque unité archivistique ou objet ou groupe d'objets est un succès (EVIDENCE_AUDIT_PREPARE_GENERATE_REPORTS.OK = Succès de la création du rapport pour chaque unité archivistique ou objet ou groupe d'objets)
  - KO : la création du rapport pour chaque unité archivistique ou objet ou groupe d'objets est un échec (EVIDENCE_AUDIT_PREPARE_GENERATE_REPORTS.KO = Échec de la création du rapport

pour chaque unité archivistique ou objet ou groupe d'objets)

- FATAL : une erreur technique est survenue de la création du rapport pour chaque unité archivistique ou objet ou groupe d'objets (EVIDENCE_AUDIT_PREPARE_GENERATE_REPORTS.FATAL = Erreur technique lors de la création du rapport pour chaque unité archivistique ou objet ou groupe d'objets)
- WARNING : avertissement lors de la création du rapport pour chaque unité archivistique ou objet ou groupe d'objets (EVIDENCE_AUDIT_PREPARE_GENERATE_REPORTS.WARNING = Avertissement lors de la création du rapport pour chaque unité archivistique ou objet ou groupe d'objets)

2.5.7. Processus de correction des signatures pour chaque objet, groupe d'objets ou unité auditée défaillante (STP_CORRECTIVE_AUDIT)

2.5.7.1. Correction des signatures pour chaque objet, groupe d'objets ou unité auditée défaillante (CORRECTIVE_AUDIT)

- **Règle** : tâche consistant à récupérer les données du/des fichiers corrompus auprès de la base de données ou d'une offre de stockage valide
- **Type** : bloquant
- **Statuts** :
  - OK : La correction des signatures de chaque unité archivistique ou objet ou groupe d'objets corrompue a bien été effectuée (CORRECTIVE_AUDIT.OK = Succès de la correction des signatures de chaque unité archivistique ou objet ou groupe d'objets corrompue)
  - KO : la correction des signatures de chaque unité archivistique ou objet ou groupe d'objets corrompue n'a pas été effectuée (CORRECTIVE_AUDIT.KO = Échec de la correction des signatures de chaque unité archivistique ou objet ou groupe d'objets corrompue)
  - FATAL : une erreur technique est survenue lors de la correction des signatures de chaque unité archivistique ou objet ou groupe d'objets corrompu (CORRECTIVE_AUDIT.FATAL = Erreur technique lors de la correction des signatures de chaque unité archivistique ou objet ou groupe d'objets corrompue)
  - WARNING : avertissement lors de la correction des signatures de chaque unité archivistique ou objet ou groupe d'objets corrompu (CORRECTIVE_AUDIT.WARNING = Avertissement lors de la correction des signatures de chaque unité archivistique ou objet ou groupe d'objets corrompue)

2.5.8. Processus de finalisation des corrections des signatures pour chaque objet, groupe d'objets ou unité auditée défaillante (STP_CORRECTION_FINALIZE)

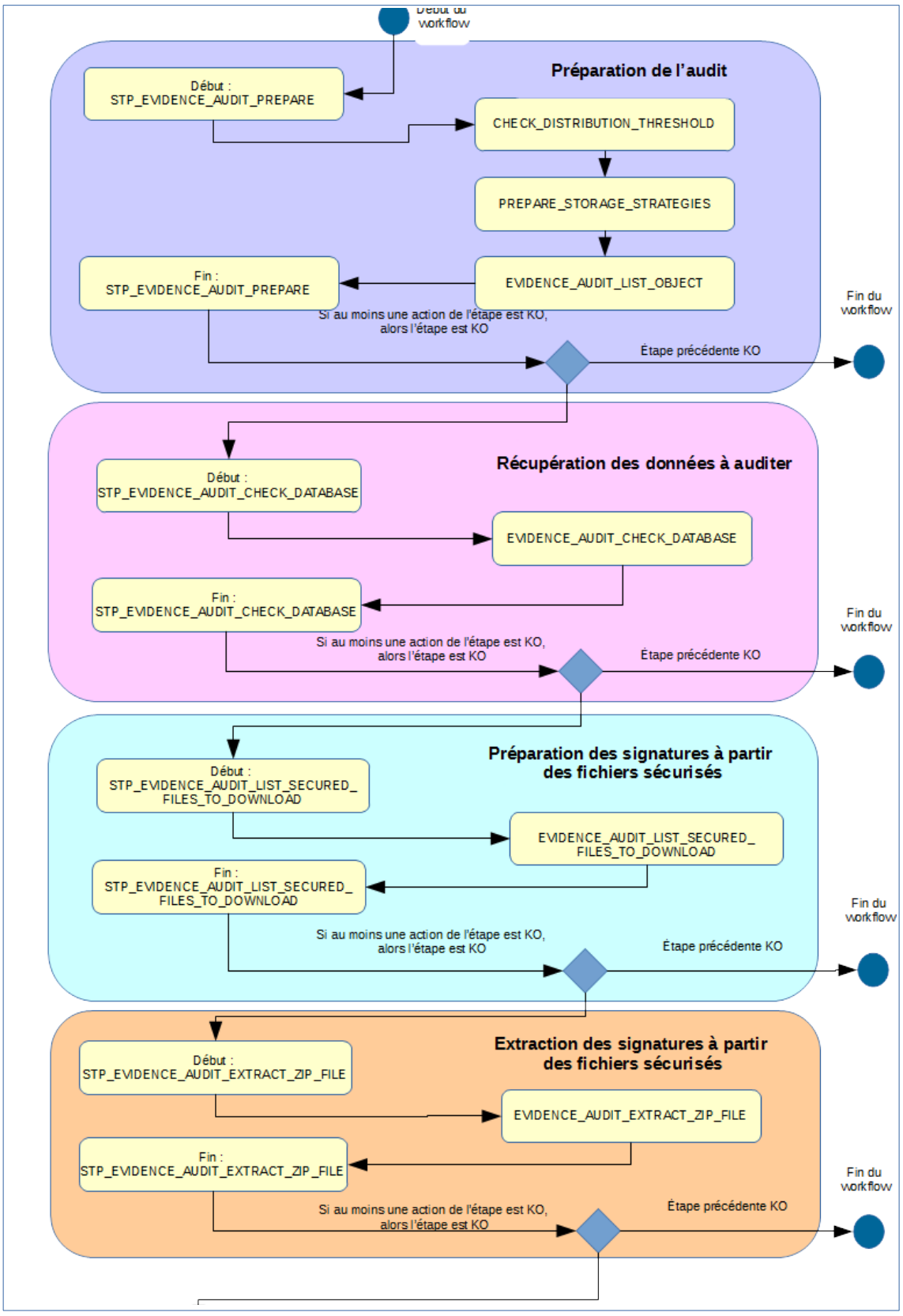
2.5.8.1. Finalisation des corrections des signatures pour chaque objet, groupe d'objets ou unité auditée, défaillante (CORRECTION_FINALIZE)

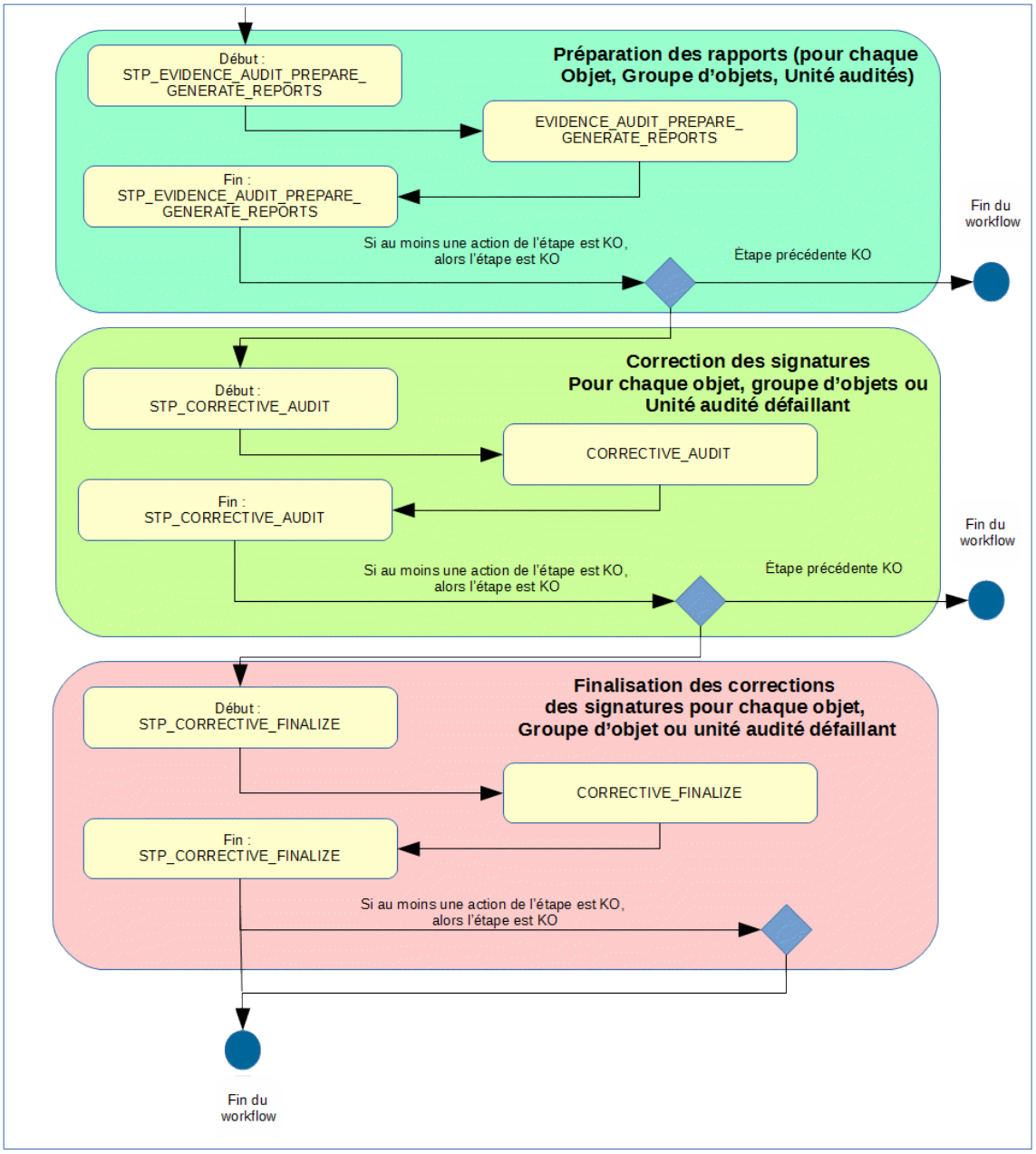
- **Règle** : tâche consistant à finaliser la correction des signatures sur les données du/des fichiers corrompus auprès de la base de données ou d'une offre de stockage valide
- **Type** : bloquant
- **Statuts** :
  - OK : la finalisation des corrections des signatures pour chaque objet, groupe d'objets ou unité auditée défaillante a bien été effectuée (CORRECTION_FINALIZE.OK = Succès du processus de finalisation des corrections des signatures pour chaque objet, groupe d'objets ou unité auditée défaillante)
  - KO : la finalisation des corrections des signatures pour chaque objet, groupe d'objets ou unité auditée défaillante n'a pas été effectuée (CORRECTION_FINALIZE.KO = Échec du processus de finalisation des corrections des signatures pour chaque objet, groupe d'objets ou unité auditée défaillante)

- FATAL : une erreur technique est survenue lors de la finalisation des corrections des signatures pour chaque objet, groupe d'objets ou unité auditée défaillante (CORRECTION_FINALIZE.FATAL = erreur technique est survenue lors du processus de finalisation des corrections des signatures pour chaque objet, groupe d'objets ou unité auditée défaillante)
- WARNING : avertissement lors de la finalisation des corrections des signatures pour chaque objet, groupe d'objets ou unité auditée défaillante (CORRECTION_FINALIZE.WARNING = Avertissement lors du processus de finalisation des corrections des signatures pour chaque objet, groupe d'objets ou unité auditée défaillante)

2.5.9.Structure du workflow de correction suite à audit

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




Préparation des rapports (pour chaque Objet, Groupe d'objets, Unité audités)
Début : STP_EVIDENCE_AUDIT_PREPARE_GENERATE_REPORTS
EVIDENCE_AUDIT_PREPARE_GENERATE_REPORTS
Fin : STP_EVIDENCE_AUDIT_PREPARE_GENERATE_REPORTS
Si au moins une action de l'étape est KO, alors l'étape est KO
Étape précédente KO
Fin du workflow
Correction des signatures Pour chaque objet, groupe d'objets ou Unité audité défaillant
Début : STP_CORRECTIVE_AUDIT
CORRECTIVE_AUDIT
Fin : STP_CORRECTIVE_AUDIT
Si au moins une action de l'étape est KO, alors l'étape est KO
Étape précédente KO
Fin du workflow
Finalisation des corrections des signatures pour chaque objet, Groupe d'objet ou unité audité défaillant
Début : STP_CORRECTIVE_FINALIZE
CORRECTIVE_FINALIZE
Fin : STP_CORRECTIVE_FINALIZE
Si au moins une action de l'étape est KO, alors l'étape est KO
Fin du workflow

2.5.10.Rapport d’audit correctif

Le rapport d’audit correctif est un fichier JSON généré par la solution logicielle Vitam lorsqu’une opération d’audit se termine. En cas de succès (RECTIFICATION_AUDIT.OK), le rapport précise uniquement l’identifiant des unités archivistiques qui ont fait l’objet de l’audit.

2.5.10.1. Exemple de JSON : rapport d’audit correctif

[{"Id":"aeaqaaaaaehgynv2aamukalojeze6iiaaaaq","Type":"UNIT"}]

2.6.Workflow de relevé de valeur probante

Cette section décrit le processus (workflow) de relevé de valeur probante. L'objectif est de rendre prouvable toute opération effectuée sur toute unité archivistique ou tout objet qui lui est associé. Ce relevé de valeur probante réunit les éléments permettant de fournir à un auditeur externe une présomption de confiance dans ce qui lui est communiqué.

Toutes les étapes, tâches et traitement sont journalisées dans le journal des opérations et décrivent le processus (clé et description de la clé associée dans le journal des opérations) tel qu'implémenté dans la version actuelle de la solution logicielle Vitam.

2.6.1.Processus de préparation du relevé de valeur probante (STP_PROBATIVE_VALUE_PREPARE)

2.6.1.1. Vérification des seuils de limitation de traitement des unités archivistiques CHECK_DISTRIBUTION_THRESHOLD

- **Règle** : étape consistant à vérifier les seuils de limitation de traitement des unités archivistiques
- **Type** : bloquant
- **Statuts** :
  - OK : la vérification des seuils de limitation de traitement des unités archivistiques a bien été effectuée (CHECK_DISTRIBUTION_THRESHOLD.OK = Succès de la vérification des seuils de limitation de traitement des unités archivistiques)
  - KO : une incohérence à été détectée entre le seuil et le nombre d'unités archivistiques à traiter. (CHECK_DISTRIBUTION_THRESHOLD.KO = Échec de la vérification des seuils de limitation de traitement des unités archivistiques)
  - FATAL : une erreur technique est survenue lors de la vérification des seuils de limitation de traitement des unités archivistiques (CHECK_DISTRIBUTION_THRESHOLD.FATAL = Erreur technique lors de la vérification des seuils de limitation de traitement des unités archivistiques)

2.6.1.2. Création du fichier de distribution permettant de réaliser le relevé de valeur probante (PROBATIVE_VALUE__CREATE_DISTRIBUTION_FILE)

- **Règle** : étape consistant à créer un fichier de distribution permettant de réaliser le relevé de valeur probante
- **Type** : bloquant
- **Statuts** :
  - OK : la création du fichier de distribution a bien effectué (PROBATIVE_VALUE_CREATE_DISTRIBUTION_FILE.OK = Succès de la création du fichier de distribution permettant de réaliser le relevé de valeur probante)
  - KO : la création du fichier de distribution n'a pas été effectué (PROBATIVE_VALUE_CREATE_DISTRIBUTION_FILE.KO = Échec lors de la création du fichier de distribution permettant de réaliser le relevé de valeur probante)
  - FATAL : une erreur technique est survenue lors de la création du fichier de distribution (PROBATIVE_VALUE_CREATE_DISTRIBUTION_FILE.FATAL = Une erreur technique est survenue lors de la création du fichier de distribution permettant de réaliser le relevé de valeur probante)

2.6.2.Processus d'alimentation du relevé de valeur probante (STP_PROBATIVE_VALUE_ACTION)

2.6.2.1. Ajout d'une entrée dans le relevé de valeur probante (PROBATIVE_VALUE_CREATE_PROBATIVE_REPORT_ENTRY)

- **Règle** : tâche consistant à créer une entrée dans le relevé de valeur probante
- **Type** : bloquant
- **Statuts** :
  - OK : la création d'une entrée dans le relevé de valeur probante à bien été effectué (PROBATIVE_VALUE_CREATE_PROBATIVE_REPORT_ENTRY.OK = Succès de la création d'une entrée dans le relevé de valeur probante)
  - KO : la création d'une entrée dans le relevé de valeur probante n'a pas été effectué (PROBATIVE_VALUE_CHECK_OBJECT_GROUP.KO = Échec de la création d'une entrée dans le relevé de valeur probante)
  - WARNING : avertissement lors de la création d'une entrée dans le relevé de valeur probante (PROBATIVE_VALUE_CHECK_OBJECT_GROUP.WARNING = Avertissement lors de la création d'une entrée dans le relevé de valeur probante)
  - FATAL : une erreur technique est survenue lors de la création d'une entrée dans le relevé de valeur probante (PROBATIVE_VALUE_CHECK_OBJECT_GROUP.FATAL = Erreur technique lors de la création d'une entrée dans le relevé de valeur probante)

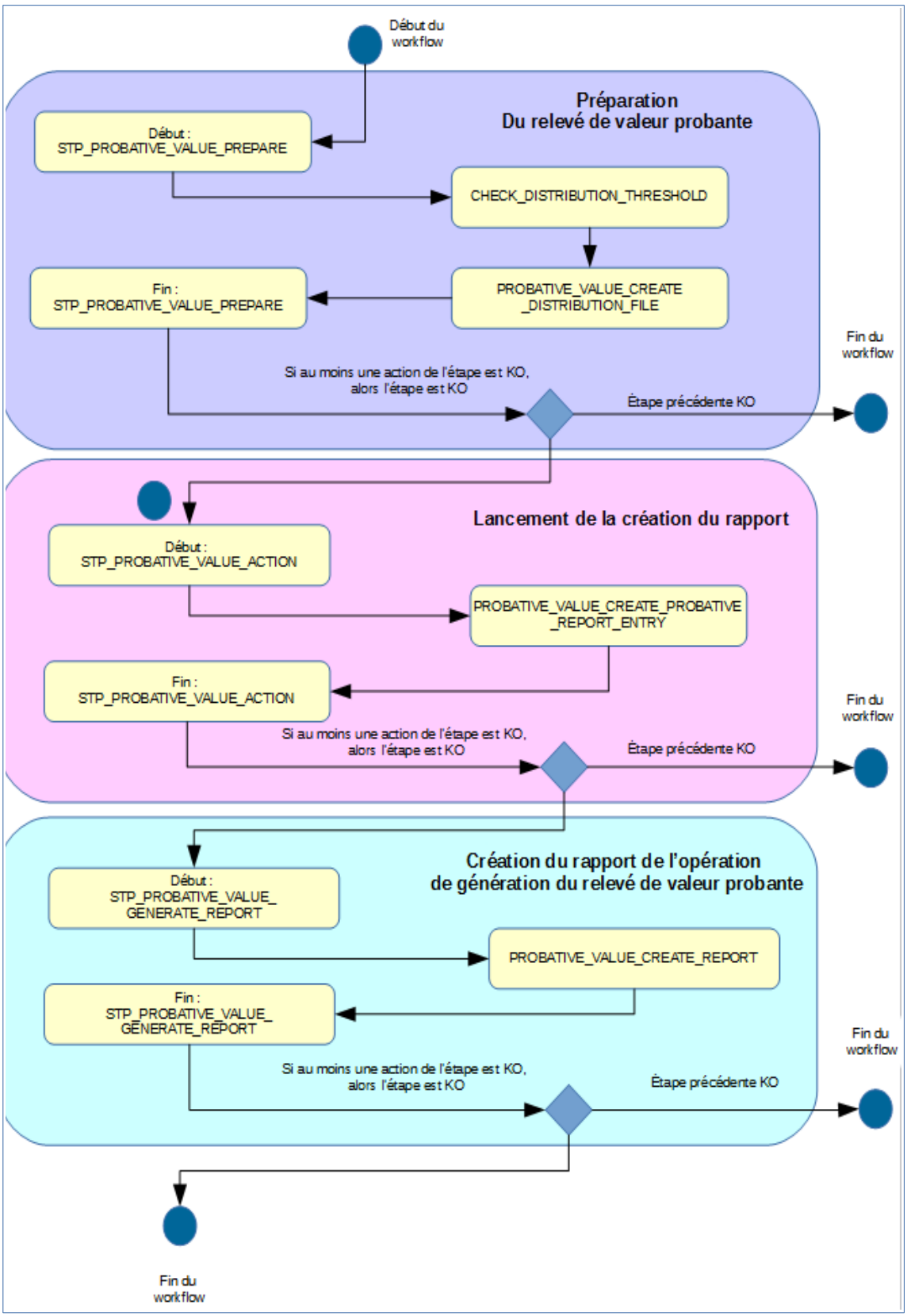
2.6.3.Processus de finalisation du relevé de valeur probante (STP_PROBATIVE_VALUE_GENERATE_REPORT)

2.6.3.1. Finalisation du relevé de valeur probante (PROBATIVE_VALUE_CREATE_REPORT)

- **Règle** : tâche consistant à finaliser le relevé de valeur probante
- **Type** : bloquant
- **Statuts** :
  - OK : Finalisation du relevé de valeur probante à été effectué avec succès. (PROBATIVE_VALUE_CREATE_REPORT.OK = Succès de la création du relevé de valeur probante pour chaque unité archivistique, objet ou groupe d'objets audité)
  - KO : Finalisation du relevé de valeur probante n'a pas été effectué. (PROBATIVE_VALUE_CREATE_REPORT.KO = Échec de la création du relevé de valeur probante pour chaque unité archivistique, objet ou groupe d'objets audité)
  - WARNING : avertissement lors de la finalisation du relevé de valeur probante (PROBATIVE_VALUE_CREATE_REPORT.WARNING = Avertissement lors de la création du relevé de valeur probante pour chaque unité archivistique, objet ou groupe d'objets audité)
  - FATAL : une erreur technique est survenue lors de la finalisation du relevé de valeur probante (PROBATIVE_VALUE_CREATE_REPORT.FATAL = Erreur technique lors de la création du relevé de valeur probante pour chaque unité archivistique, objet ou groupe d'objets audité)

2.6.4.Structure de workflow du relevé de valeur probante

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



2.7.Rapport du relevé de valeur probante

Le relevé de valeur probante est un fichier JSON généré par la solution logicielle Vitam. Le relevé de valeur probante réunit les éléments permettant de fournir à un auditeur externe une présomption de confiance dans ce qui lui est communiqué.

2.7.1.Exemple de JSON : rapport de valeur probante

OperationSummary

ReportSummary

Context

```
{
  "operationSummary" : {
    "tenant" : 0,
    "evId" : "aeeaaaaaachemhquabmugaloiuua7xaaaaaq",
    "evType" : "EXPORT_PROBATIVE_VALUE",
    "outcome" : "OK",
    "outDetail" : "EXPORT_PROBATIVE_VALUE.OK",
    "outMsg" : "Succès lors du processus d'export du relevé de valeur probante",
    "rightsStatementIdentifier" : {
      "AccessContract" : "ContratINR"
    }
  },

  "reportSummary" : {
    "evStartDateTime" : "2019-11-07T09:18:05.178",
    "evEndDateTime" : "2019-11-07T09:18:06.640",
    "reportType" : "PROBATIVE_VALUE",
    "vitamResults" : {
      "OK" : 1,
      "KO" : 0,
      "WARNING" : 0,
      "total" : 1
    }
  },

  "context" : {
    "dslQuery" : {
      "$query" : [ {
        "$or" : [ {
          "$in" : {
            "#id" : [ "aeaqaaaaaahgynv2aa3oealoiqpvbraaaaba" ]
          }
```

```
        }, {
          "$in" : {
            "#allunitups" : [ ]
          }
        } ]
      } ],
      "$filter" : { },
      "$projection" : { }
    },
    "usage" : "BinaryMaster",
    "version" : "1"
  }

  "reportEntries" : [ {
    "unitIds" : [ "aeaqaaaaaahgynv2aa3oealoiqpvbraaaaba" ],
    "objectGroupId" : "aebaaaaaaahgynv2aa3oealoiqpva2qaaaaq",
    "objectId" : "aeaaaaaaaahgynv2aa3oealoiqpvazyaaaaq",
    "usageVersion" : "BinaryMaster_1",
    "operations" : [ {
      "id" : "aecaaaaaachnaqi7abbwqaloirpkzpiaaaaq",
      "evTypeProc" : "STP_OP_SECURISATION",
      "evDateTime" : "2019-11-07T05:38:05.629"
    }, {
      "id" : "aecaaaaaachnaqi7abbwqaloirpsv3qaaaaq",
      "evTypeProc" : "LOGBOOK_OBJECTGROUP_LFC_TRACEABILITY",
      "evDateTime" : "2019-11-07T05:38:37.934"
    }, {
      "id" : "aeeaaaaaachemhquabwdcaloiqpuvhiaaaaq",
      "evTypeProc" : "PROCESS_SIP_UNITARY",
      "rightsStatementIdentifier" : "{\"ArchivalAgreement\":\"ArchivalAgreement0\"}",
      "agIdApp" : "CT-000001",
      "evDateTime" : "2019-11-07T04:28:51.829"
    } ],
    "checks" : [ {
      "name" : "TIMESTAMP_OPERATION_DATABASE_TRACEABILITY_VALIDATION",
      "details" : "Validating timestamp operation found in database 'AND' timestamp
operation found in traceability secured file.",
      "type" : "TIMESTAMP_CHECKING",
```


ReportDetail

```
      "source" : "DATABASE",

      "destination" : "TRACEABILITY_FILE",

      "sourceComparable" :
"MIILITAVAgEAMBAMDk9wZXJhdGlvbiBPa2F5MIILBgYJKoZIhvcNAQcCoIIK9zCCCvMCAQMxDzANBglghkgBZQM
EAgMFADCBgAYLKoZIhvcNAQkQAQSgcQRvMG0CAQEGASkwUTANBglghkgBZQMEAgMFAARA4fpDituDLaS2w6jnrd3
1fi/
4VyJQ9uzEyVvngUM2IybKkVjZzTNLwnO3lZb+ZHrjvYoc2+2gzJ8Rw93deFzAtgIBARgPMjAxOTExMDcwNTM4MDd
aoIIGhzCCBoMwggRroAMCAQICAgDUMA0GCSqGSIb3DQEBCwUAMHgxCzAJBgNVBAYTAmZyMQwwCgYDVQQIDANPZGY
xDjAMBgNVBAcMBXBhcmlzMQ4wDAYDVQQKDAV2aXRhbTEUMBIGA1UECwwLYXV0aG9yaXRpZXMxJTAjBgNVBAMMHGN
hX2ludGVybWVkaWF0ZV90aW1lc3RhbXBpbmcwHhcNMTkxMTA3MDIzMzA1WhcNMjIxMTA2MDIzMzA1WjBUMQswCQY
DVQQGEwJmcjEMMAoGA1UECAwDaWRmMQ4wDAYDVQQHDAVwYXJpczEOMAwGA1UECgwFdml0YW0xFzAVBgNVBAMMDnN
lY3VyZS1sb2dib29rMIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEA09lqSj5nvYlemZC4CHCzTpHNgK1
WzA/ILRjVL4ho2vfT6yx9TstAQqsVARrsknmAGJ2KMillffIOEZ00ph3H0oo7auWOuGDytsfteRMdFSVOxUfM/
1jBUXgqhg6VSHVka1ueAu0kg4g0ADvWrJbtVKHxps9vrEA+H4tEiTT3mfKEdWBAyLz4cGOln4ip3s1GMSjmVgDwV
gbzPnHvD1VUDLij6dYHbkAQaLSRh9hl47dPVbzOe1qZAUPDML6kVUI6fM9jxAzQJqub0jL+vEGvMUmJTqSTnFXAU
C5GOf3LU5iV/cwyGUWY0rz5kKS1XLT9jvYteijxYe4nf0y/V5Yzp/6F/
52esT4rqI5x9QGfv9TxSHiymcu316au14uFw+M+qLy4khC1BhNbsHojjnGhR33KuRa/
QAZltPJ3KyFC3NLSfGrJSi+JbZV5yS5yybkd1F25TdyUSFxk+0lQEvmL6ngsjKQHqSNdsTWmAHdBuE614ErrySxG
wjnFBMOfsPK2HKe3dc7iZUZYhPPUyBq8RVWDvHVFtQuXBCkCMSVNK6rVjmYMuSiI8oAy0TK7ixfAqMmE2yZwoJt7
hMYbo0Adrfjg7Bf95uP2LDC8LkWsALhIJdFOJT5ANXsWxGM0LmFmGTRoPFOb2E0G/
QQKAc0GEgRiG4qGihCLiZSGJ81CYw0CAwEAAaOCATkwggE1MCUGCWCGSAGG+EIBDQQYFhZDZXJ0aWZpY2F0IFNlc
nZldXIgU1NMMB0GA1UdDgQWBBRpel2gJ6UBuGBr8LG5d7Sa8kfmazCBmwYDVR0jBIGTMIGQgBQHzKjBQYoL+UCSH
cYIsFk+rST006F0pHIwcDELMAkGA1UEBhMCZnIxDDAKBgNVBAgMA2lkZjEOMAwGA1UEBwwFcGFyaXMxDjAMBgNVB
AoMBXZpdGFtMRQwEgYDVQQLDAthdXRob3JpdGllczEdMBsGA1UEAwwUY2Ffcm9vdF90aW1lc3RhbXBpbmeCAgDTM
AkGA1UdEgQCMAAwDAYDVR0TAQH/BAIwADALBgNVHQ8EBAMCBsAwEQYJYIZIAYb4QgEBBAQDAgZAMBYGA1UdJQEB/
wQMMAoGCCsGAQUFBwMIMA0GCSqGSIb3DQEBCwUAA4ICAQAwjDb7PvGAmqgW62+mXQHHo6hh8ikJyyz6kT10jLaRV
Uou77ZMoUqF5GlS6d+tSpvGL5cdrs2Z5WqZPny3DyOxBBsGnIY9FM0herQAr6mY6xyTB+TrQDfqZeaqL0m1I/
PAWiQNwjDUJCDGEhvaoOm046fJNKcXrPjb9cQ/
CEsWdOJ4ERBZnUV8nOyVjCPQeW2yrY1hZrtYrfpTGl7thy0INGBz/zySHl5ArF1lbD75Z3o9ABMtRTqPjJ5R/
FeEzOYXZdpdc5C7UqQwyZXcQpRQlYJvedBDrBqGQ5AZ9KMnQEmQEHoE1zyxWoxpo2I1C/G/X/
nvBIiqgVUgsxTdh/btOIQPvKS/PuvmuqUeCb1KKwS7txgY4g5387q8NuV

MrhUc6iux3J1mcX7ismaAk/1AZPojzbNNxFwW5N/
4+q2TM8Q+0q3lxO9wyiGQqoCuP76ZvO0REukpFgUsBf+Yot1qyTxFYxpwgW1hX9ed8VsWJPHUNOy97Cd8MOPfgj+
CefBemvMa8zQH958KWtkjLlYKAhQuXHM5E6CuDUpE98090+ulNp4g7lSSJBaLOCMZWplji0syFRIGuirNloDruhi
lX/NOuGvBk5AUtvVvsbfFm3gOCpx+ijUMLcaBd5e6J0+xyvK2v0a+/x25hPjuLp/
weXmiMGtANyzbgmvJrNkgUDGCA80wggPJAgEBMH4weDELMAkGA1UEBhMCZnIxDDAKBgNVBAgMA2lkZjEOMAwGA1U
EBwwFcGFyaXMxDjAMBgNVBAoMBXZpdGFtMRQwEgYDVQQLDAthdXRob3JpdGllczElMCMGA1UEAwwcY2FfaW50ZXJ
tZWRpYXRlX3RpbWVzdGFtcGluZwICANQwDQYJYIZIAWUDBAIDBQCgggEgMBoGCSqGSIb3DQEJAzENBgsqhkiG9w0
BCRABBDAcBgkqhkiG9w0BCQUxDxcNMTkxMTA3MDUzODA3WjAtBgkqhkiG9w0BCTQxIDAeMA0GCWCGSAFlAwQCAwU
AoQ0GCSqGSIb3DQEBDQUAME8GCSqGSIb3DQEJBDFCBECZFLQlPJunF7t9ueEP12tiDJ31dsEAWDnL/
6UUUi+duet24AvfKjtOOlwlEeJrR0oHeYnUadXZwGQo7/+Qs5a0MGQGCyqGSIb3DQEJEAIvMVUwUzBRME8wCwYJY
IZIAWUDBAIDBEAeduVLSR3MT88jYzzAMR8nbaiHoZPSUfMPxK/
1Ekh3DXK7z46X0bKIs87kWN8Rjacz7QuF7nDQmPRcETZHjUjoMA0GCSqGSIb3DQEBDQUABIICAF6U4lAIlcQTEJG
OXIQinnZ4clRer9SW91s9WDWzCxyOq4d60vnq3rTgKdNiIYBeLomRujr3DjUI6Byt57mHJe/
AWqvAb+1BBhUWkPKjbWadkXPyeMxMTHhgUUKd4OBMMBITK0g9WWR7Vz5ouB2AOV2XMcKce5B17N5Rlv5mH5J9EgH
Ffp+osLE2GM5jtw5L8nOxA+U9056vxXvKRMNCKk8X6cKGJCL9uUkq+UiZQu1sQEAlQC+7x6YMvfL51RXy+oRzD5W
85InR5IqGFjjdazD1wbONBODrVFn4t0PI6PqW56KFsJDYXhP4xiyY9s+O857blP3rMfiDrPDCEcJzTc6ChOmn2XD
/5F6958w9x/3LqOCI1c7grLLuCk28ndgiS5nuQ4k/
yOEnLbGhlZuFKDxpAZctRCS3gDz4hLcbJmiCns0trQ29BybHddwzqUQf2LHuzrciZgWjpB+MYOBTfUu+ifau8pSB
1wctAe6GSczKDYS/E0tLd9NFzXayxDwgslqvxdV6FdstLGb6LHe8xFNf/KfNzgzeQN8K6xE76qc3/
dvqmIKUWqzndBjc/
6tl3zDxOBnyFITcnCWJZFTLbCPA+ZUaw4yiWC26Gz7BxOUtTvurZoyB5VkCdMHb4IQdrR1GhG1hRH9v7ES8dBHcZ
dHnflX5igjqPJrEpZVDeZfM",

      "destinationComparable" :
"MIILITAVAgEAMBAMDk9wZXJhdGlvbiBPa2F5MIILBgYJKoZIhvcNAQcCoIIK9zCCCvMCAQMxDzANBglghkgBZQM
EAgMFADCBgAYLKoZIhvcNAQkQAQSgcQRvMG0CAQEGASkwUTANBglghkgBZQMEAgMFAARA4fpDituDLaS2w6jnrd3
1fi/
4VyJQ9uzEyVvngUM2IybKkVjZzTNLwnO3lZb+ZHrjvYoc2+2gzJ8Rw93deFzAtgIBARgPMjAxOTExMDcwNTM4MDd
aoIIGhzCCBoMwggRroAMCAQICAgDUMA0GCSqGSIb3DQEBCwUAMHgxCzAJBgNVBAYTAmZyMQwwCgYDVQQIDANPZGY
xDjAMBgNVBAcMBXBhcmlzMQ4wDAYDVQQKDAV2aXRhbTEUMBIGA1UECwwLYXV0aG9yaXRpZXMxJTAjBgNVBAMMHGN
hX2ludGVybWVkaWF0ZV90aW1lc3RhbXBpbmcwHhcNMTkxMTA3MDIzMzA1WhcNMjIxMTA2MDIzMzA1WjBUMQswCQY
DVQQGEwJmcjEMMAoGA1UECAwDaWRmMQ4wDAYDVQQHDAVwYXJpczEOMAwGA1UECgwFdml0YW0xFzAVBgNVBAMMDnN
lY3VyZS1sb2dib29rMIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEA09lqSj5nvYlemZC4CHCzTpHNgK1
WzA/ILRjVL4ho2vfT6yx9TstAQqsVARrsknmAGJ2KMillffIOEZ00ph3H0oo7auWOuGDytsfteRMdFSVOxUfM/
1jBUXgqhg6VSHVka1ueAu0kg4g0ADvWrJbtVKHxps9vrEA+H4tEiTT3mfKEdWBAyLz4cGOln4ip3s1GMSjmVgDwV
```

gbzPnHvD1VUDLij6dYHbkAQaLSRh9hl47dPVbzOe1qZAUPDML6kVUI6fM9jxAzQJqub0jL+vEGvMUmJTqSTnFXAU
C5GOf3LU5iV/cwyGUWY0rz5kKS1XLT9jvYteijxYe4nf0y/V5Yzp/6F/
52esT4rqI5x9QGfv9TxSHiymcu316au14uFw+M+qLy4khC1BhNbsHojjnGhR33KuRa/
QAZltPJ3KyFC3NLSfGrJSi+JbZV5yS5yybkd1F25TdyUSFxk+0lQEvmL6ngsjKQHqSNdsTWmAHdBuE614ErrySxG
wjnFBMOfsPK2HKe3dc7iZUZYhPPUyBq8RVWDvHVFtQuXBCkCMSVNK6rVjmYMuSiI8oAy0TK7ixfAqMmE2yZwoJt7
hMYbo0Adrfjg7Bf95uP2LDC8LkWsALhIJdFOJT5ANXsWxGM0LmFmGTRoPFOb2E0G/
QQKAc0GEgRiG4qGihCLiZSGJ81CYw0CAwEAAaOCATkwggE1MCUGCWCGSAGG+EIBDQQYFhZDZXJ0aWZpY2F0IFNlc
nZldXIgU1NMMB0GA1UdDgQWBBRpel2qJ6UBuGBr8LG5d7Sa8kfmazCBmwYDVR0jBIGTMIGOgBQHzKjBOYoL+UCSH
cYIsFk+rST006F0pHIwcDELMAkGA1UEBhMCZnIxDDAKBgNVBAgMA2lkZjEOMAwGA1UEBwwFcGFyaXMxDjAMBgNVB
AoMBXZpdGFtMRQwEgYDVQQLDAthdXRob3JpdGllczEdMBsGA1UEAwwUY2Ffcm9vdF90aW1lc3RhbXBpbmeCAgDTM
AkGA1UdEgQCMAAwDAYDVR0TAQH/BAIwADALBgNVHQ8EBAMCBsAwEQYJYIZIAYb4QgEBBAQDAgZAMBYGA1UdJQEB/
wQMMAoGCCsGAQUFBwMIMA0GCSqGSIb3DQEBCwUAA4ICAQAwjDb7PvGAmqgW62+mXQHHo6hh8ikJyyz6kT10jLaRV
Uou77ZMoUqF5GlS6d+tSpvGL5cdrs2Z5WqZPny3DyOxBBsGnIY9FM0herQAr6mY6xyTB+TrQDfqZeaqL0m1I/
PAWiQNwjDUJCDGEhvaoOm046fJNKcXrPjb9cQ/
CEsWdOJ4ERBZnUV8nOyVjCPQeW2yrY1hZrtYrfpTGl7thy0INGBz/zySHl5ArF1lbD75Z3o9ABMtRTqPjJ5R/
FeEzOYXZdpdc5C7UqQwyZXcQpRQlYJvedBDrBqGQ5AZ9KMnQEmQEHoE1zyxWoxpo2I1C/G/X/
nvBIiqgVUgsxTdh/btOIQPvKS/PuvmuqUeCb1KKwS7txgY4g5387q8NuVMrhUc6iux3J1mcX7ismaAk/
1AZPojzbNNxFwW5N/
4+q2TM8Q+0q3lxO9wyiGQqoCuP76ZvO0REukpFgUsBf+Yot1qyTxFYxpwgW1hX9ed8VsWJPHUNOy97Cd8MOPfgj+
CefBemvMa8zQH958KWtkjLlYKAhQuXHM5E6CuDUpE98090+ulNp4g7lSSJBaLOCMZWplji0syFRIGuirNloDruhi
lX/NOuGvBk5AUtvVvsbfFm3gOCpx+ijUMLcaBd5e6J0+xyvK2v0a+/x25hPjuLp/
weXmiMGtANyzbgmvJrNkgUDGCA80wggPJAgEBMH4weDELMAkGA1UEBhMCZnIxDDAKBgNVBAgMA2lkZjEOMAwGA1U
EBwwFcGFyaXMxDjAMBgNVBAoMBXZpdGFtMRQwEgYDVQQLDAthdXRob3JpdGllczElMCMGA1UEAwwcY2FfaW50ZXJ
tZWRpYXRlX3RpbWVzdGFtcGluZwICANQwDQYJYIZIAWUDBAIDBQCgggEgMBoGCSqGSIb3DQEJAzENBgsqhkiG9w0
BCRABBDAcBgkqhkiG9w0BCQUxDxcNMTkxMTA3MDUzODA3WjAtBgkqhkiG9w0BCTQxIDAeMA0GCWCGSAFlAwQCAwU
AoQ0GCSqGSIb3DQEBDQUAME8GCSqGSIb3DQEJBDFCBECZFLQlPJunF7t9ueEP12tiDJ31dsEAWDnL/
6UUUi+duet24AvfKjtOOlwlEeJrR0oHeYnUadXZwGQo7/+Qs5a0MGQGCyqGSIb3DQEJEAIvMVUwUzBRME8wCwYJY
IZIAWUDBAIDBEAeduVLSR3MT88jYzzAMR8nbaiHoZPSUfMPxK/
1Ekh3DXK7z46X0bKIs87kWN8Rjacz7QuF7nDQmPRcETZHjUjoMA0GCSqGSIb3DQEBDQUABIICAF6U4lAIlcQTEJG
OXIQinnZ4clRer9SW91s9WDWzCxyOq4d60vnq3rTgKdNiIYBeLomRujr3DjUI6Byt57mHJe/
AWqvAb+1BBhUWkPKjbWadkXPyeMxMTHhgUUKd4OBMMBITK0g9WWR7Vz5ouB2AOV2XMcKce5B17N5Rlv5mH5J9EgH
Ffp+osLE2GM5jtw5L8nOxA+U9056vxXvKRMNCKk8X6cKGJCL9uUkq+UiZQu1sQEAlQC+7x6YMvfL51RXy+oRzD5W
85InR5IqGFjjdazD1wbONBODrVFn4t0PI6PqW56KFsJDYXhP4xiyY9s+O857blP3rMfiDrPDCEcJzTc6ChOmn2XD
/5F6958w9x/3LqOCI1c7grLLuCk28ndgiS5nuQ4k/
yOEnLbGhlZuFKDxpAZctRCS3gDz4hLcbJmiCns0trQ29BybHddwzqUQf2LHuzrciZgWjpB+MYOBTfUu+ifau8pSB
1wctAe6GSczKDYS/E0tLd9NFzXayxDwgslqvxdV6FdstLGb6LHe8xFNf/KfNzgzeQN8K6xE76qc3/
dvqmIKUWqzndBjc/
6tl3zDxOBnyFITcnCWJZFTLbCPA+ZUaw4yiWC26Gz7BxOUtTvurZoyB5VkCdMHb4IQdrR1GhG1hRH9v7ES8dBHcZ
dHnflX5igjqPJrEpZVDeZfM",

      "action" : "VALIDATION",

      "item" : "TIMESTAMP_OPERATION",

      "status" : "OK"

    }, {

      "name" : "TIMESTAMP_OPERATION_DATABASE_TRACEABILITY_COMPARISON",

      "details" : "Comparing timestamp operation found in database with timestamp
operation found in traceability secured file.",

      "type" : "TIMESTAMP_CHECKING",

      "source" : "DATABASE",

      "destination" : "TRACEABILITY_FILE",

      "sourceComparable" :
"MIILITAVAgEAMBAMDk9wZXJhdGlvbiBPa2F5MIILBgYJKoZIhvcNAQcCoIIK9zCCCvMCAQMxDzANBglghkgBZQM
EAgMFADCBgAYLKoZIhvcNAQkQAQSgcQRvMG0CAQEGASkwUTANBglghkgBZQMEAgMFAARA4fpDituDLaS2w6jnrd3
1fi/
4VyJQ9uzEyVvngUM2IybKkVjZzTNLwnO3lZb+ZHrjvYoc2+2gzJ8Rw93deFzAtgIBARgPMjAxOTExMDcwNTM4MDd
aoIIGhzCCBoMwggRroAMCAQICAgDUMA0GCSqGSIb3DQEBCwUAMHgxCzAJBgNVBAYTAmZyMQwwCgYDVQQIDANpZGY
xDjAMBgNVBAcMBXBhcmlzMQ4wDAYDVQQKDAV2aXRhbTEUMBIGA1UECwwLYXV0aG9yaXRpZXMxJTAjBgNVBAMMHGN
hX2ludGVybWVkaWF0ZV90aW1lc3RhbXBpbmcwHhcNMTkxMTA3MDIzMzA1WhcNMjIxMTA2MDIzMzA1WjBUMQswCQY
DVQQGEwJmcjEMMAoGA1UECAwDaWRmMQ4wDAYDVQQHDAVwYXJpczEOMAwGA1UECgwFdml0YW0xFzAVBgNVBAMMDnN
lY3VyZS1sb2dib29rMIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEA09lqSj5nvYlemZC4CHCzTpHNgK1
WzA/ILRjVL4ho2vfT6yx9TstAQqsVARrsknmAGJ2KMillffIOEZ00ph3H0oo7auWOuGDytsfteRMdFSVOxUfM/

```
1jBUXgqhg6VSHVka1ueAu0kg4g0ADvWrJbtVKHxps9vrEA+H4tEiTT3mfKEdWBAyLz4cGOln4ip3s1GMSjmVgDwV
gbzPnHvD1VUDLij6dYHbkAQaLSRh9hl47dPVbzOe1qZAUPDML6kVUI6fM9jxAzQJqub0jL+vEGvMUmJTqSTnFXAU
C5GOf3LU5iV/cwyGUWY0rz5kKS1XLT9jvYteijxYe4nf0y/V5Yzp/6F/
52esT4rqI5x9QGfv9TxSHiymcu316au14uFw+M+qLy4khC1BhNbsHojjnGhR33KuRa/
QAZltPJ3KyFC3NLSfGrJSi+JbZV5yS5yybkd1F25TdyUSFxk+0lQEvmL6ngsjKQHqSNdsTWmAHdBuE614ErrySxG
wjnFBMOfsPK2HKe3dc7iZUZYhPPUyBq8RVWDvHVFtQuXBCkCMSVNK6rVjmYMuSiI8oAy0TK7ixfAqMmE2yZwoJt7
hMYbo0Adrfjg7Bf95uP2LDC8LkWsALhIJdFOJT5ANXsWxGM0LmFmGTRoPFOb2E0G/
QQKAc0GEgRiG4qGihCLiZSGJ81CYw0CAwEAAaOCATkwggE1MCUGCWCGSAGG+EIBDQQYFhZDZXJ0aWZpY2F0IFNlc
nZldXIgU1NMMB0GA1UdDgQWBBRpel2gJ6UBuGBr8LG5d7Sa8kfmazCBmwYDVR0jBIGTMIGQgBQHzKjBQYoL+UCSH
cYIsFk+rST006F0pHIwcDELMAkGA1UEBhMCZnIxDDAKBgNVBAgMA2lkZjEOMAwGA1UEBwwFcGFyaXMxDjAMBgNVB
AoMBXZpdGFtMRQwEgYDVQQLDAthdXRob3JpdGllczEdMBsGA1UEAwwUY2Ffcm9vdF90aW1lc3RhbXBpbmeCAgDTM
AkGA1UdEgQCMAAwDAYDVR0TAQH/BAIwADALBgNVHQ8EBAMCBsAwEQYJYIZIAYb4QgEBBAQDAgZAMBYGA1UdJQEB/
wQMMAoGCCsGAQUFBwMIMA0GCSqGSIb3DQEBCwUAA4ICAQAwjDb7PvGAmqgW62+mXQHHo6hh8ikJyyz6kT10jLaRV
Uou77ZMoUqF5GlS6d+tSpvGL5cdrs2Z5WqZPny3DyOxBBsGnIY9FM0herQAr6mY6xyTB+TrQDfqZeaqL0m1I/
PAWiQNwjDUJCDGEhvaoOm046fJNKcXrPjb9cQ/
CEsWdOJ4ERBZnUV8nOyVjCPQeW2yrY1hZrtYrfpTGl7thy0INGBz/zySHl5ArF1lbD75Z3o9ABMtRTqPjJ5R/
FeEzOYXZdpdc5C7UqQwyZXcQpRQlYJvedBDrBqGQ5AZ9KMnQEmQEHoE1zyxWoxpo2I1C/G/X/
nvBIiqgVUgsxTdh/btOIQPvKS/PuvmuqUeCb1KKwS7txgY4g5387q8NuVMrhUc6iux3J1mcX7ismaAk/
1AZPojzbNNxFwW5N/
4+q2TM8Q+0q3lxO9wyiGQqoCuP76ZvO0REukpFgUsBf+Yot1qyTxFYxpwgW1hX9ed8VsWJPHUNOy97Cd8MOPfgj+
CefBemvMa8zQH958KWtkjLlYKAhQuXHM5E6CuDUpE98090+ulNp4g7lSSJBaLOCMZWplji0syFRIGuirNloDruhi
lX/NOuGvBk5AUtvVvsbfFm3gOCpx+ijUMLcaBd5e6J0+xyvK2v0a+/x25hPjuLp/
weXmiMGtANyzbgmvJrNkgUDGCA80wggPJAgEBMH4weDELMAkGA1UEBhMCZnIxDDAKBgNVBAgMA2lkZjEOMAwGA1U
EBwwFcGFyaXMxDjAMBgNVBAoMBXZpdGFtMRQwEgYDVQQLDAthdXRob3JpdGllczElMCMGA1UEAwwcY2FfaW50ZXJ
tZWRpYXRlX3RpbWVzdGFtcGluZwICANQwDQYJYIZIAWUDBAIDBQCgggEgMBoGCSqGSIb3DQEJAzENBgsqhkiG9w0
BCRABBDAcBgkqhkiG9w0BCQUxDxcNMTkxMTA3MDUzODA3WjAtBgkqhkiG9w0BCTQxIDAeMA0GCWCGSAFlAwQCAwU
AoQ0GCSqGSIb3DQEBDQUAME8GCSqGSIb3DQEJBDFCBECZFLQ1PJunF7t9ueEP12tiDJ31dsEAWDnL/
6UUUi+duet24AvfKjtOOlwlEeJrR0oHeYnUadXZwGQo7/+Qs5a0MGQGCyqGSIb3DQEJEAIvMVUwUzBRME8wCwYJY
IZIAWUDBAIDBEAeduVLSR3MT88jYzzAMR8nbaiHoZPSUfMPxK/
1Ekh3DXK7z46X0bKIs87kWN8Rjacz7QuF7nDQmPRcETZHjUjoMA0GCSqGSIb3DQEBDQUABIICAF6U4lAIlcQTEJG
OXIQinnZ4clRer9SW91s9WDWzCxyOq4d60vnq3rTgKdNiIYBeLomRujr3DjUI6Byt57mHJe/
AWqvAb+1BBhUWkPKjbWadkXPyeMxMTHhgUUKd4OBMMBITK0g9WWR7Vz5ouB2AOV2XMcKce5B17N5Rlv5mH5J9EgH
Ffp+osLE2GM5jtw5L8nOxA+U9056vxXvKRMNCKk8X6cKGJCL9uUkq+UiZQu1sQEAlQC+7x6YMvfL51RXy+oRzD5W
85InR5IqGFjjdazD1wbONBODrVFn4t0PI6PqW56KFsJDYXhP4xiyY9s+O857blP3rMfiDrPDCEcJzTc6ChOmn2XD
/5F6958w9x/3LqOCI1c7grLLuCk28ndgiS5nuQ4k/
yOEnLbGhlZuFKDxpAZctRCS3gDz4hLcbJmiCns0trQ29BybHddwzqUQf2LHuzrciZgWjpB+MYOBTfUu+ifau8pSB
1wctAe6GSczKDYS/E0tLd9NFzXayxDwgslqvxdV6FdstLGb6LHe8xFNf/KfNzgzeQN8K6xE76qc3/
dvqmIKUWqzndBjc/
6tl3zDxOBnyFITcnCWJZFTLbCPA+ZUaw4yiWC26Gz7BxOUtTvurZoyB5VkCdMHb4IQdrR1GhG1hRH9v7ES8dBHcZ
dHnflX5igjqPJrEpZVDeZfM",

      "destinationComparable" :
"MIILITAVAgEAMBAMDk9wZXJhdGlvbiBPa2F5MIILBgYJKoZIhvcNAQcCoIIK9zCCCvMCAQMxDzANBglghkgBZQM
EAgMFADCBgAYLKoZIhvcNAQkQAQSgcQRvMG0CAQEGASkwUTANBglghkgBZQMEAgMFAARA4fpDituDLaS2w6jnrd3
1fi/
4VyJQ9uzEyVvngUM2IybKkVjZzTNLwnO3lZb+ZHrjvYoc2+2gzJ8Rw93deFzAtgIBARgPMjAxOTExMDcwNTM4MDd
aoIIGhzCCBoMwggRroAMCAQICAgDUMA0GCSqGSIb3DQEBCwUAMHgxCzAJBgNVBAYTAmZyMQwwCgYDVQQIDANpZGY
xDjAMBgNVBAcMBXBhcmlzMQ4wDAYDVQQKDAV2aXRhbTEUMBIGA1UECwwLYXV0aG9yaXRpZXMxJTAjBgNVBAMMHGN
hX2ludGVybWVkaWF0ZV90aW1lc3RhbXBpbmcwHhcNMTkxMTA3MDIzMzA1WhcNMjIxMTA2MDIzMzA1WjBUMQswCQY
DVQQGEwJmcjEMMAoGA1UECAwDaWRmMQ4wDAYDVQQHDAVwYXJpczEOMAwGA1UECgwFdml0YW0xFzAVBgNVBAMMDnN
lY3VyZS1sb2dib29rMIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEA09lqSj5nvYlemZC4CHCzTpHNgK1
WzA/ILRjVL4ho2vfT6yx9TstAQqsVARrsknmAGJ2KMillffIOEZ00ph3H0oo7auWOuGDytsfteRMdFSVOxUfM/
1jBUXgqhg6VSHVka1ueAu0kg4g0ADvWrJbtVKHxps9vrEA+H4tEiTT3mfKEdWBAyLz4cGOln4ip3s1GMSjmVgD

wVgbzPnHvD1VUDLij6dYHbkAQaLSRh9hl47dPVbzOe1qZAUPDML6kVUI6fM9jxAzQJqub0jL+vEGvMUmJTqSTnFX
AUC5GOf3LU5iV/cwyGUWY0rz5kKS1XLT9jvYteijxYe4nf0y/V5Yzp/6F/
52esT4rqI5x9QGfv9TxSHiymcu316au14uFw+M+qLy4khC1BhNbsHojjnGhR33KuRa/
QAZltPJ3KyFC3NLSfGrJSi+JbZV5yS5yybkd1F25TdyUSFxk+0lQEvmL6ngsjKQHqSNdsTWmAHdBuE614ErrySxG
wjnFBMOfsPK2HKe3dc7iZUZYhPPUyBq8RVWDvHVFtQuXBCkCMSVNK6rVjmYMuSiI8oAy0TK7ixfAqMmE2yZwoJt7
hMYbo0Adrfjg7Bf95uP2LDC8LkWsALhIJdFOJT5ANXsWxGM0LmFmGTRoPFOb2E0G/
QQKAc0GEgRiG4qGihCLiZSGJ81CYw0CAwEAAaOCATkwggE1MCUGCWCGSAGG+EIBDQQYFhZDZXJ0aWZpY2F0IFNlc
nZldXIgU1NMMB0GA1UdDgQWBBRpel2gJ6UBuGBr8LG5d7Sa8kfmazCBmwYDVR0jBIGTMIGQgBQHzKjBQYoL+UCSH
cYIsFk+rST006F0pHIwcDELMAkGA1UEBhMCZnIxDDAKBgNVBAgMA2lkZjEOMAwGA1UEBwwFcGFyaXMxDjAMBgNVB
AoMBXZpdGFtMRQwEgYDVQQLDAthdXRob3JpdGllczEdMBsGA1UEAwwUY2Ffcm9vdF90aW1lc3RhbXBpbmeCAgDTM
AkGA1UdEgQCMAAwDAYDVR0TAQH/BAIwADALBgNVHQ8EBAMCBsAwEQYJYIZIAYb4QgEBBAQDAgZAMBYGA1UdJQEB/
wQMMAoGCCsGAQUFBwMIMA0GCSqGSIb3DQEBCwUAA4ICAQAwjDb7PvGAmqgW62+mXQHHo6hh8ikJyyz6kT10jLaRV
Uou77ZMoUqF5GlS6d+tSpvGL5cdrs2Z5WqZPny3DyOxBBsGnIY9FM0herQAr6mY6xyTB+TrQDfqZeaqL0m1I/
PAWiQNwjDUJCDGEhvaoOm046fJNKcXrPjb9cQ/
CEsWdOJ4ERBZnUV8nOyVjCPQeW2yrY1hZrtYrfpTGl7thy0INGBz/zySHl5ArF1lbD75Z3o9ABMtRTqPjJ5R/
```

FeEzOYXZdpdc5C7UqQwyZXcQpRQlYJvedBDrBqGQ5AZ9KMnQEmQEHoE1zyxWoxpo2I1C/G/X/
nvBIiqgVUgsxTdh/btOIQPvKS/PuvmuqUeCb1KKwS7txgY4g5387q8NuVMrhUc6iux3J1mcX7ismaAk/
1AZPojzbNNxFwW5N/
4+q2TM8Q+0q3lxO9wyiGQqoCuP76ZvO0REukpFgUsBf+Yot1qyTxFYxpwgW1hX9ed8VsWJPHUNOy97Cd8MOPfgj+
CefBemvMa8zQH958KWtkjLlYKAhQuXHM5E6CuDUpE98090+ulNp4g7lSSJBaLOCMZWplji0syFRIGuirNloDruhi
lX/NOuGvBk5AUtvVvsbfFm3gOCpx+ijUMLcaBd5e6J0+xyvK2v0a+/x25hPjuLp/
weXmiMGtANyzbgmvJrNkgUDGCA80wggPJAgEBMH4weDELMAkGA1UEBhMCZnIxDDAKBgNVBAgMA2lkZjEOMAwGA1U
EBwwFcGFyaXMxDjAMBgNVBAoMBXZpdGFtMROwEgYDVQQLDAthdXRob3JpdGllczElMCMGA1UEAwwcY2FfaW50ZXJ
tZWRpYXRlX3RpbWVzdGFtcGluZwICANQwDQYJYIZIAWUDBAIDBQCgggEgMBoGCSqGSIb3DQEJAzENBgsqhkiG9w0
BCRABBDAcBgkqhkiG9w0BCQUxDxcNMTkxMTA3MDUzODA3WjAtBgkqhkiG9w0BCTQxIDAeMA0GCWCGSAFlAwQCAwU
AoQ0GCSqGSIb3DQEBDQUAME8GCSqGSIb3DQEJBDFCBECZFLQlPJunF7t9ueEP12tiDJ31dsEAWDnL/
6UUUi+duet24AvfKjtOOlwlEeJrR0oHeYnUadXZwGQo7/+Qs5a0MGQGCyqGSIb3DQEJEAIvMVUwUzBRME8wCwYJY
IZIAWUDBAIDBEAeduVLSR3MT88jYzzAMR8nbaiHoZPSUfMPxK/
1Ekh3DXK7z46X0bKIs87kWN8Rjacz7QuF7nDQmPRcETZHjUjoMA0GCSqGSIb3DQEBDQUABIICAF6U4lAIlcQTEJG
OXIQinnZ4clRer9SW91s9WDWzCxyOq4d60vnq3rTgKdNiIYBeLomRujr3DjUI6Byt57mHJe/
AWqvAb+1BBhUWkPKjbWadkXPyeMxMTHhgUUKd4OBMMBITK0g9WWR7Vz5ouB2AOV2XMcKce5B17N5Rlv5mH5J9EgH
Ffp+osLE2GM5jtw5L8nOxA+U9056vxXvKRMNCKk8X6cKGJCL9uUkq+UiZQu1sQEAlQC+7x6YMvfL51RXy+oRzD5W
85InR5IqGFjjdazD1wbONBODrVFn4t0PI6PqW56KFsJDYXhP4xiyY9s+O857blP3rMfiDrPDCEcJzTc6ChOmn2XD
/5F6958w9x/3LqOCI1c7grLLuCk28ndgiS5nuQ4k/
yOEnLbGhlZuFKDxpAZctRCS3gDz4hLcbJmiCns0trQ29BybHddwzqUQf2LHuzrciZgWjpB+MYOBTfUu+ifau8pSB
1wctAe6GSczKDYS/E0tLd9NFzXayxDwgslqvxdV6FdstLGb6LHe8xFNf/KfNzgzeQN8K6xE76qc3/
dvqmIKUWqzndBjc/
6tl3zDxOBnyFITcnCWJZFTLbCPA+ZUaw4yiWC26Gz7BxOUtTvurZoyB5VkCdMHb4IQdrR1GhG1hRH9v7ES8dBHcZ
dHnflX5igjqPJrEpZVDeZfM",

```
      "action" : "COMPARISON",

      "item" : "TIMESTAMP_OPERATION",

      "status" : "OK"

    }, {

      "name" : "MERKLE_OPERATION_DIGEST_DATABASE_TRACEABILITY_COMPARISON",

      "details" : "Comparing operation merkle digest found in database with operation
merkle digest found in traceability secured file.",

      "type" : "MERKLE_INTEGRITY",

      "source" : "DATABASE",

      "destination" : "TRACEABILITY_FILE",

      "sourceComparable" :
"JI89QHGLjTANLgYByeJ2NY21JOQTMnL3UOmvRICW7Oe8e+0fXSK6CPyLDXXnt1/9KyphZaKXODZvgnrV+PsJww=
=",

      "destinationComparable" :
"JI89QHGLjTANLgYByeJ2NY21JOQTMnL3UOmvRICW7Oe8e+0fXSK6CPyLDXXnt1/9KyphZaKXODZvgnrV+PsJww=
=",

      "action" : "COMPARISON",

      "item" : "MERKLE_TREE_ROOT_OPERATION_DIGEST",

      "status" : "OK"

    }, {

      "name" : "MERKLE_OPERATION_DIGEST_COMPUTATION_TRACEABILITY_COMPARISON",

      "details" : "Comparing operation merkle digest computed from secured data with
operation merkle digest found in traceability secured file.",

      "type" : "MERKLE_INTEGRITY",

      "source" : "COMPUTATION",

      "destination" : "TRACEABILITY_FILE",

      "sourceComparable" :
```

```
"JI89QHGLjTANLgYByeJ2NY21JOQTMnL3UOmvRICW7Oe8e+0fXSK6CPyLDXXnt1/9KyphZaKXODZvgnrV+PsJww=
=",

      "destinationComparable" :
"JI89QHGLjTANLgYByeJ2NY21JOQTMnL3UOmvRICW7Oe8e+0fXSK6CPyLDXXnt1/9KyphZaKXODZvgnrV+PsJww=
=",

      "action" : "COMPARISON",

      "item" : "MERKLE_TREE_ROOT_OPERATION_DIGEST",

      "status" : "OK"

    }, {

      "name" : "MERKLE_OPERATION_DIGEST_COMPUTATION_ADDITIONAL_TRACEABILITY_COMPARISON",

      "details" : "Comparing operation merkle digest computed from secured data with
operation merkle digest found in additional traceability secured file.",

      "type" : "MERKLE_INTEGRITY",

      "source" : "COMPUTATION",

      "destination" : "ADDITIONAL_TRACEABILITY",

      "sourceComparable" :
"JI89QHGLjTANLgYByeJ2NY21JOQTMnL3UOmvRICW7Oe8e+0fXSK6CPyLDXXnt1/9KyphZaKXODZvgnrV+PsJww=
=",

      "destinationComparable" :
"JI89QHGLjTANLgYByeJ2NY21JOQTMnL3UOmvRICW7Oe8e+0fXSK6CPyLDXXnt1/9KyphZaKXODZvgnrV+PsJww=
=",

      "action" : "COMPARISON",

      "item" : "MERKLE_TREE_ROOT_OPERATION_DIGEST",

      "status" : "OK"

    }, {

      "name" : "PREVIOUS_TIMESTAMP_OPERATION_DATABASE_TRACEABILITY_VALIDATION",

      "details" : "Validating previous timestamp operation found in database 'AND'
previous timestamp operation found in traceability secured file.",

      "type" : "CHAIN",

      "source" : "DATABASE",

      "destination" : "TRACEABILITY_FILE",

      "sourceComparable" :
"MIILITAVAgEAMBAMDk9wZXJhdGlvbiBPa2F5MIILBgYJKoZIhvcNAQcCoIIK9zCCCvMCAQMxDzANBglghkgBZQM
EAgMFADCBgAYLKoZIhvcNAQkQAQSgcQRvMG0CAQEGASkwUTANBglghkgBZQMEAgMFAARACPaKQZakyyhRDnsAhDW
afWC2tL1A1Wg0AZIk2ycySfEFppfp1hY6Zpy0c82fZ7srdKPl+uWyqxH6a5q5lDbl5AIBARgPMjAxOTExMDcwNDA
wMDlaoIIGhzCCBoMwggRroAMCAQICAgDUMA0GCSqGSIb3DQEBCwUAMHgxCzAJBgNVBAYTAmZyMQwwCgYDVQQIDAN
pZGYxDjAMBgNVBAcMBXBhcmlzMQ4wDAYDVQQKDAV2aXRhbTEUMBIGA1UECwwLYXV0aG9yaXRpZXMxJTAjBgNVBAM
MHGNhX2ludGVybWVkaWF0ZV90aW1lc3RhbXBpbmcwHhcNMTkxMTA3MDIzMzA1WhcNMjIxMTA2MDIzMzA1WjBUMQs
wCQYDVQQGEwJmcjEMMAoGA1UECAwDaWRmMQ4wDAYDVQQHDAVwYXJpczEOMAwGA1UECgwFdml0YW0xFzAVBgNVBAM
MDnNlY3VyZS1sb2dib29rMIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEA09lqSj5nvYlemZC4CHCzTpH
NgK1WzA/
ILRjVL4ho2vfT6yx9TstAQqsVARrsknmAGJ2KMillffIOEZ00ph3H0oo7auWOuGDytsfteRMdFSVOxUfM/
1jBUXgqhg6VSHVka1ueAu0kg4g0ADvWrJbtVKHxps9vrEA+H4tEiTT3mfKEdWBAyLz4cGOln4ip3s1GMSjmVgDwV
gbzPnHvD1VUDLij6dYHbkAQaLSRh9hl47dPVbzOe1qZAUPDML6kVUI6fM9jxAzQJqub0jL+vEGvMUmJTqSTnFXAU
C5GOf3LU5iV/cwyGUWY0rz5kKS1XLT9jvYteijxYe4nf0y/V5Yzp/6F/
52esT4rqI5x9QGfv9TxSHiymcu316au14uFw+M+qLy4khC1BhNbsHojjnGhR33KuRa/
QAZltPJ3KyFC3NLSfGrJSi+JbZV5yS5yybkd1F25TdyUSFxk+0lQEvmL6ngsjKQHqSNdsTWmAHdBuE614ErrySxG
wjnFBMOfsPK2HKe3dc7iZUZYhPPUyBq8RVWDvHVFtQuXBCkCMSVNK6rVjmYMuSiI8oAy0TK7ixfAqMmE2yZwoJt7
hMYbo0Adrfjg7Bf95uP2LDC8LkWsALhIJdFOJT5ANXsWxGM0LmFmGTRoPFOb2E0G/
QQKAc0GEgRiG4qGihCLiZSGJ81CYw0CAwEAAaOCATkwggE1MCUGCWCGSAGG+EIBDQQYFhZDZXJ0aWZpY2F0IFNlc
```

nZldXIgU1NMMB0GA1UdDgQWBBRpel2gJ6UBuGBr8LG5d7Sa8kfmazCBmwYDVR0jBIGTMIGQgBQHzKjBQYoL+UCSH
cYIsFk+rST006F0pHIwcDELMAkGA1UEBhMCZnIxDDAKBgNVBAgMA2lkZjEOMAwGA1UEBwwFcGFyaXMxDjAMBgNVB
AoMBXZpdGFtMRQwEgYDVQQLDAthdXRob3JpdGllczEdMBsGA1UEAwwUY2Ffcm9vdF90aW1lc3RhbXBpbmeCAgDTM
AkGA1UdEgQCMAAwDAYDVR0TAQH/BAIwADALBgNVHQ8EBAMCBsAwEQYJYIZIAYb4QgEBBAQDAgZAMBYGA1UdJQEB/
wQMMAoGCCsGAQUFBwMIMA0GCSqGSIb3DQEBCwUAA4ICAQAwjDb7PvGAmqgW62+mXQHHo6hh8ikJyyz6kT10jLaRV
Uou77ZMoUqF5GlS6d+tSpvGL5cdrs2Z5WqZPny3DyOxBBsGnIY9FM0herQAr6mY6xyTB+TrQDfqZeaqL0m1I/
PAWiQNwjDUJCDGEhvaoOm046fJNKcXrPjb9cQ/
CEsWdOJ4ERBZnUV8nOyVjCPQeW2yrY1hZrtYrfpTGl7thy0INGBz/zySHl5ArF1lbD75Z3o9ABMtRTqPjJ5R/
FeEzOYXZdpdc5C7UqQwyZXcQpRQlYJvedBDrBqGQ5AZ9KMnQEmQEHoE1zyxWoxpo2I1C/G/X/nvBIiqgVUgsxTdh/btOIQPvKS/
PuvmuqUeCb1KKwS7txgY4g5387q8NuVMrhUc6iux3J1mcX7ismaAk/1AZPojzbNNxFwW5N/
4+q2TM8Q+0q3lxO9wyiGQqoCuP76ZvO0REukpFgUsBf+Yot1qyTxFYxpwgW1hX9ed8VsWJPHUNOy97Cd8MOPfgj+
CefBemvMa8zQH958KWtkjLlYKAhQuXHM5E6CuDUpE98090+ulNp4g7lSSJBaLOCMZWplji0syFRIGuirNloDruhi
lX/NOuGvBk5AUtvVvsbfFm3gOCpx+ijUMLcaBd5e6J0+xyvK2v0a+/x25hPjuLp/
weXmiMGtANyzbgmvJrNkgUDGCA80wggPJAgEBMH4weDELMAkGA1UEBhMCZnIxDDAKBgNVBAgMA2lkZjEOMAwGA1U
EBwwFcGFyaXMxDjAMBgNVBAoMBXZpdGFtMRQwEgYDVQQLDAthdXRob3JpdGllczElMCMGA1UEAwwcY2FfaW50ZXJ
tZWRpYXRlX3RpbWVzdGFtcGluZwICANQwDQYJYIZIAWUDBAIDBQCgggEgMBoGCSqGSIb3DQEJAzENBgsqhkiG9w0
BCRABBDAcBgkqhkiG9w0BCQUxDxcNMTkxMTA3MDQwMDA5WjAtBgkqhkiG9w0BCTQxIDAeMA0GCWCGSAFlAwQCAwU
AoQ0GCSqGSIb3DQEBDQUAME8GCSqGSIb3DQEJBDFCBECaBL12eV9W7T0Y2Qbm43JWJDlmeXFeRgiwCTSfh91uIPd
xavc8huFSy5UuP9xeJ8gX7/
NQ5ftuNy4SHxRM60YyMGQGCyqGSIb3DQEJEAIvMVUwUzBRME8wCwYJYIZIAWUDBAIDBEAeduVLSR3MT88jYzzAMR
8nbaiHoZPSUfMPxK/
1Ekh3DXK7z46X0bKIs87kWN8Rjacz7QuF7nDQmPRcETZHjUjoMA0GCSqGSIb3DQEBDQUABIICALZUqEjXV7rYVWA
gJ31YxJHADF3mYOv3aehjQlGMIuCzzmdzgW9V3FagQvQB60rJg/sApdwgvRM4uAX9TG0sVKigsZ/
sxhnhEEg5CEdP93R16y47/
TZYXs37Vqa3Qycd5xiEvL4zmJUl2ZA2SKptweiqt3VmWID4cZkYonTgwTgFIwev3YpDXd/
1Dy1zV5AnAs65bO37jVqxjVRUA2PUwxAvvOxG6uQLXj7GlbncTaQpSYhMErR1H0lpNvQ1F3D3O/
RpVSWnVjbToLTeADDsk4fJjsIU1fZeC3Im/
OZw1rKVzEDZdtuRvCr7LIst1JtxEcZTHDgiKmQXN4t39h+S49nBOldFq/
V43IdqMT+bjKxQlOG0x4dtnIvfDnH0eu93KuHr9ivGDp3GnDBLzePTX99kzM2Ky1vBeVL2+MzN27CIdceuZlQ1zp
ahXDgi+iOERhIf1GS+P2iR4vF4GKbUvsYQUrB2WNNaG/
naffGFDPVcd4tB6rJDOBR1TzgwaSEwUamlVebSkZBgvDIalzB12CuefhY+bMUPGbtQlWrrr6HizPTL2/
KrPakCoZk5nw9/
MICO+20eU6gP+s69TsV32CCTIpKxK87QD0NHz2ZBj5Ejte4h1sYxtR+3HvQYITaHlYxHMvy0KZVytMqCMVM/
Fyf2LGN9XjPd+ddDvyA7htRT",

"destinationComparable" :
"MIILITAVAgEAMBAMDk9wZXJhdGlvbiBPa2F5MIILBgYJKoZIhvcNAQcCoIIK9zCCCvMCAQMxDzANBglghkgBZQM
EAgMFADCBgAYLKoZIhvcNAQkQAQSgcQRvMG0CAQEGASkwUTANBglghkgBZQMEAgMFAARACPaKQZakyyhRDnsAhDW
afWC2tL1A1Wg0AZIk2ycySfEFppfp1hY6Zpy0c82fZ7srdKPl+uWyqxH6a5q5lDbl5AIBARgPMjAxOTExMDcwNDA
wMDlaoIIGhzCCBoMwggRroAMCAQICAgDUMA0GCSqGSIb3DQEBCwUAMHgxCzAJBgNVBAYTAmZyMQwwCgYDVQQIDAN
pZGYxDjAMBgNVBAcMBXBhcmlzMQ4wDAYDVQQKDAV2aXRhbTEUMBIGA1UECwwLYXV0aG9yaXRpZXMxJTAjBgNVBAM
MHGNhX2ludGVybWVkaWF0ZV90aW1lc3RhbXBpbmcwHhcNMTkxMTA3MDIzMzA1WhcNMjIxMTA2MDIzMzA1WjBUMQs
wCQYDVQQGEwJmcjEMMAoGA1UECAwDaWRmMQ4wDAYDVQQHDAVwYXJpczEOMAwGA1UECgwFdml0YW0xFzAVBgNVBAM
MDnNlY3VyZS1sb2dib29rMIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEA09lqSj5nvYlemZC4CHCzTpH
NgK1WzA/
ILRjVL4ho2vfT6yx9TstAQqsVARrsknmAGJ2KMillffIOEZ00ph3H0oo7auWOuGDytsfteRMdFSVOxUfM/
1jBUXgqhg6VSHVka1ueAu0kg4g0ADvWrJbtVKHxps9vrEA+H4tEiTT3mfKEdWBAyLz4cGOln4ip3s1GMSjmVgDwV
gbzPnHvD1VUDLij6dYHbkAQaLSRh9hl47dPVbzOe1qZAUPDML6kVUI6fM9jxAzQJqub0jL+vEGvMUmJTqSTnFXAU
C5GOf3LU5iV/cwyGUWY0rz5kKS1XLT9jvYteijxYe4nf0y/V5Yzp/6F/
52esT4rqI5x9QGfv9TxSHiymcu316au14uFw+M+qLy4khC1BhNbsHojjnGhR33KuRa/
QAZltPJ3KyFC3NLSfGrJSi+JbZV5yS5yybkd1F25TdyUSFxk+0lQEvmL6ngsjKQHqSNdsTWmAHdBuE614ErrySxG
wjnFBMOfsPK2HKe3dc7iZUZYhPPUyBq8RVWDvHVFtQuXBCkCMSVNK6rVjmYMuSiI8oAy0TK7ixfAqMmE2yZwoJt7
hMYbo0Adrfjg7Bf95uP2LDC8LkWsALhIJdFOJT5ANXsWxGM0LmFmGTRoPFOb2E0G/
QQKAc0GEgRiG4qGihCLiZSGJ81CYw0CAwEAAaOCATkwggE1MCUGCWCGSAGG+EIBDQQYFhZDZXJ0aWZpY2F0IFNlc
nZldXIgU1NMMB0GA1UdDgQWBBRpel2gJ6UBuGBr8LG5d7Sa8kfmazCBmwYDVR0jBIGTMIGQgBQHzKjBQYoL+UCSH
cYIsFk+rST006F0pHIwcDELMAkGA1UEBhMCZnIxDDAKBgNVBAgMA2lkZjEOMAwGA1UEBwwFcGFyaXMxDjAMBgNVB
AoMBXZpdGFtMRQwEgYDVQQLDAthdXRob3JpdGllczEdMBsGA1UEAwwUY2Ffcm9vdF90aW1lc3RhbXBpbmeCAgDTM
AkGA1UdEgQCMAAwDAYDVR0TAQH/BAIwADALBgNVHQ8EBAMCBsAwEQYJYIZIAYb4QgEBBAQDAgZAMBYGA1UdJQEB/
wQMMAoGCCsGAQUFBwMIMA0GCSqGSIb3DQEBCwUAA4ICAQAwjDb7PvGAmqgW62+mXQHHo6hh8ikJyyz6kT10jLaRV
Uou77ZMoUqF5GlS6d+tSpvGL5cdrs2Z5WqZPny3DyOxBBsGnIY9FM0herQAr6mY6xyTB+TrQDfqZeaqL0m1I/
PAWiQNwjDUJCDGEhvaoOm046fJNKcXrPjb9cQ/
CEsWdOJ4ERBZnUV8nOyVjCPQeW2yrY1hZrtYrfpTGl7thy0INGBz/zySHl5ArF1lbD75Z3o9ABMtRTqPjJ5R/
FeEzOYXZdpdc5C7UqQwyZXcQpRQlYJvedBDrBqGQ5AZ9KMnQEmQEHoE1zyxWoxpo2I1C/G/X/
nvBIiqgVUgsxTdh/btOIQPvKS/PuvmuqUeCb1KKwS7txgY4g5387q8NuVMrhUc6iux3J1mcX7ismaAk/
1AZPojzbNNxFwW5N/
4+q2TM8Q+0q3lxO9wyiGQqoCuP76ZvO0REukpFgUsBf+Yot1qyTxFYxpwgW1hX9ed8VsWJPHUNOy97Cd8MOPfgj+
CefBemvMa8zQH958KWtkjLlYKAhQuXHM5E6CuDUpE98090+ulNp4g7lSSJBaLOCMZWplji0syFRIGuirNloDruhi

lX/NOuGvBk5AUtvVvsbfFm3gOCpx+ijUMLcaBd5e6J0+xyvK2v0a+/x25hPjuLp/
weXmiMGtANyzbgmvJrNkgUDGCA80wggPJAgEBMH4weDELMAkGA1UEBhMCZnIxDDAKBgNVBAgMA2lkZjEOMAwGA1U
EBwwFcGFyaXMxDjAMBgNVBAoMBXZpdGFtMRQwEgYDVQQLDAthdXRob3JpdGllczElMCMGA1UEAwwcY2FfaW50ZXJ
tZWRpYXRlX3RpbWVzdGFtcGluZwICANQwDQYJYIZIAWUDBAIDBQCgggEgMBoGCSqGSIb3DQEJAzENBgsqhkiG9w0
BCRABBDAcBgkqhkiG9w0BCQUxDxcNMTkxMTA3MDQwMDA5WjAtBgkqhkiG9w0BCTQxIDAeMA0GCWCGSAFlAwQCAwU
AoQ0GCSqGSIb3DQEBDQUAME8GCSqGSIb3DQEJBDFCBECaBL12eV9W7T0Y2Qbm43JWJDlmeXFeRgiwCTSfh91uIPd
xavc8huFSy5UuP9xeJ8gX7/
NQ5fuNy4SHxPM60YyMCQCCyqGSIb3DQEJFAIxMVUxUzBPME8wCwYJYIZIAWUDBAIDBEAodwVLSD3MT88jYzzAMD
8nbaiHoZPSUfMPxK/
1Ekh3DXK7z46X0bKIs87kWN8Rjacz7QuF7nDQmPRcETZHjUjoMA0GCSqGSIb3DQEBDQUABIICALZUqEjXV7rYVWA
gJ31YxJHADF3mYOv3aehjQlGMIuCzzmdzgW9V3FagQvQB60rJg/sApdwgvRM4uAX9TG0sVKigsZ/
sxhnhEEg5CEdP93R16y47/
TZYXs37Vqa3Qycd5xiEvL4zmJUl2ZA2SKptweiqt3VmWID4cZkYonTgwTgFIwev3YpDXd/
1Dy1zV5AnAs65bO37jVqxjVRUA2PUwxAvvOxG6uQLXj7GlbncTaQpSYhMErR1H0lpNvQ1F3D3O/
RpVSWnVjbToLTeADDsk4fJjsIU1fZeC3Im/
OZw1rKVzEDZdtuRvCr7LIst1JtxEcZTHDgiKmQXN4t39h+S49nBOldFq/
V43IdqMT+bjKxQlOG0x4dtnIvfDnH0eu93KuHr9ivGDp3GnDBLzePTX99kzM2Ky1vBeVL2+MzN27CIdceuZlQ1zp
ahXDgi+iOERhIf1GS+P2iR4vF4GKbUvsYQUrB2WNNaG/
naffGFDPVcd4tB6rJDOBR1TzgwaSEwUamlVebSkZBgvDIalzB12CuefhY+bMUPGbtQlWrrr6HizPTL2/
KrPakCoZk5nw9/
MICO+20eU6gP+s69TsV32CCTIpKxK87QD0NHz2ZBj5Ejte4h1sYxtR+3HvQYITaHlYxHMvy0KZVytMqCMVM/
Fyf2LGN9XjPd+ddDvyA7htRT",

      "action" : "VALIDATION",

      "item" : "PREVIOUS_TIMESTAMP_OPERATION",

      "status" : "OK"

    }, {

      "name" : "TIMESTAMP_OPERATION_COMPUTATION_TRACEABILITY_COMPARISON",

      "details" : "Comparing timestamp operation computed from computing information
traceability file with timestamp operation found in traceability secured file.",

      "type" : "TIMESTAMP_CHECKING",

      "source" : "COMPUTATION",

      "destination" : "TRACEABILITY_FILE",

      "sourceComparable" :
"4fpDituDLaS2w6jnrd31fi/4VyJQ9uzEyVvngUM2IybKkVjZzTNLwnO3lZb+ZHrjvYoc2+2gzJ8Rw93deFzAtg=
=",

      "destinationComparable" :
"4fpDituDLaS2w6jnrd31fi/4VyJQ9uzEyVvngUM2IybKkVjZzTNLwnO3lZb+ZHrjvYoc2+2gzJ8Rw93deFzAtg=
=",

      "action" : "COMPARISON",

      "item" : "TIMESTAMP_OPERATION",

      "status" : "OK"

    }, {

      "name" : "PREVIOUS_TIMESTAMP_OPERATION_DATABASE_TRACEABILITY_COMPARISON",

      "details" : "Comparing previous timestamp operation found in database with
previous timestamp operation found in traceability secured file.",

      "type" : "CHAIN",

      "source" : "DATABASE",

      "destination" : "TRACEABILITY_FILE",

      "sourceComparable" :
"MIILITAVAgEAMBAMDk9wZXJhdGlvbiBPa2F5MIILBgYJKoZIhvcNAQcCoIIK9zCCCvMCAQMxDzANBglghkgBZQM
EAgMFADCBgAYLKoZIhvcNAQkQAQSgcQRvMG0CAQEGASkwUTANBglghkgBZQMEAgMFAARACPaKQZakyyhRDnsAhDW

```
afWC2tL1A1Wg0AZIk2ycySfEFppfp1hY6Zpy0c82fZ7srdKPl+uWyqxH6a5q5lDbl5AIBARgPMjAxOTExMDcwNDA
wMDlaoIIGhzCCBoMwggRroAMCAQICAgDUMA0GCSqGSIb3DQEBCwUAMHgxCzAJBgNVBAYTAmZyMQwwCgYDVQQIDAN
pZGYxDjAMBgNVBAcMBXBhcmlzMQ4wDAYDVQQKDAV2aXRhbTEUMBIGA1UECwwLYXV0aG9yaXRpZXMxJTAjBgNVBAM
MHGNhX2ludGVybWVkaWF0ZV90aW1lc3RhbXBpbmcwHhcNMTkxMTA3MDIzMzA1WhcNMjIxMTA2MDIzMzA1WjBUMQs
wCQYDVQQGEwJmcjEMMAoGA1UECAwDaWRmMQ4wDAYDVQQHDAVwYXJpczEOMAwGA1UECgwFdml0YW0xFzAVBgNVBAM
MDnNlY3VyZS1sb2dib29rMIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEA09lqSj5nvYlemZC4CHCzTpH
NgK1WzA/
ILRjVL4ho2vfT6yx9TstAQqsVARrsknmAGJ2KMillffIOEZ00ph3H0oo7auWOuGDytsfteRMdFSVOxUfM/
1jBUXgqhg6VSHVka1uEAu0kg4g0ADvWrJbtVKHxps9vrEA+H4tEiTT3mfKEdWBAyLz4cGOln4ip3s1GMSjmVgDwV
gbzPnHvD1VUDLij6dYHbkAQaLSRh9hl47dPVbzOe1qZAUPDML6kVUI6fM9jxAzQJqub0jL+vEGvMUmJTqSTnFXAU
C5GOf3LU5iV/cwyGUWY0rz5kKS1XLT9jvYteijxYe4nf0y/V5Yzp/6F/
52esT4rqI5x9QGfv9TxSHiymcu316au14uFw+M+qLy4khC1BhNbsHojjnGhR33KuRa/
QAZltPJ3KyFC3NLSfGrJSi+JbZV5yS5yybkd1F25TdyUSFxk+0lQEvmL6ngsjKQHqSNdsTWmAHdBuE614ErrySxG
wjnFBMOfsPK2HKe3dc7iZUZYhPPUyBq8RVWDvHVFtQuXBCkCMSVNK6rVjmYMuSiI8oAy0TK7ixfAqMmE2yZwoJt7
hMYbo0Adrfjg7Bf95uP2LDC8LkWsALhIJdFOJT5ANXsWxGM0LmFmGTRoPFOb2E0G/
QQKAc0GEgRiG4qGihCLiZSGJ81CYw0CAwEAAaOCATkwggE1MCUGCWCGSAGG+EIBDQQYFhZDZXJ0aWZpY2F0IFNlc
nZldXIgU1NMMB0GA1UdDgQWBBRpel2gJ6UBuGBr8LG5d7Sa8kfmazCBmwYDVR0jBIGTMIGQgBQHzKjBQYoL+UCSH
cYIsFk+rST006F0pHIwcDELMAkGA1UEBhMCZnIxDDAKBgNVBAgMA2lkZjEOMAwGA1UEBwwFcGFyaXMxDjAMBgNVB
AoMBXZpdGFtMRQwEgYDVQQLDAthdXRob3JpdGllczEdMBsGA1UEAwwUY2Ffcm9vdF90aW1lc3RhbXBpbmeCAgDTM
AkGA1UdEgQCMAAwDAYDVR0TAQH/BAIwADALBgNVHQ8EBAMCBsAwEQYJYIZIAYb4QgEBBAQDAgZAMBYGA1UdJQEB/
wQMMAoGCCsGAQUFBwMIMA0GCSqGSIb3DQEBCwUAA4ICAQAwjDb7PvGAmqgW62+mXQHHo6hh8ikJyyz6kT10jLaRV
Uou77ZMoUqF5GlS6d+tSpvGL5cdrs2Z5WqZPny3DyOxBBsGnIY9FM0herQAr6mY6xyTB+TrQDfqZeaqL0m1I/
PAWiQNwjDUJCDGEhvaoOm046fJNKcXrPjb9cQ/
CEsWdOJ4ERBZnUV8nOyVjCPQeW2yrY1hZrtYrfpTGl7thy0INGBz/zySHl5ArF1lbD75Z3o9ABMtRTqPjJ5R/
FeEzOYXZdpdc5C7UqQwyZXcQpRQlYJvedBDrBqGQ5AZ9KMnQEmQEHoE1zyxWoxpo2I1C/G/X/
nvBIiqgVUgsxTdh/btOIQPvKS/PuvmuqUeCb1KKwS7txgY4g5387q8NuVMrhUc6iux3J1mcX7ismaAk/
1AZPojzbNNxFwW5N/
4+q2TM8Q+0q3lxO9wyiGQqoCuP76ZvO0REukpFgUsBf+Yot1qyTxFYxpwgW1hX9ed8VsWJPHUNOy97Cd8MOPfgj+
CefBemvMa8zQH958KWtkjLlYKAhQuXHM5E6CuDUpE98090+ulNp4g7lSSJBaLOCMZWplji0syFRIGuirNloDruhi
lX/NOuGvBk5AUtvVvsbfFm3gOCpx+ijUMLcaBd5e6J0+xyvK2v0a+/x25hPjuLp/
weXmiMGtANyzbgmvJrNkgUDGCA80wggPJAgEBMH4weDELMAkGA1UEBhMCZnIxDDAKBgNVBAgMA2lkZjEOMAwGA1U
EBwwFcGFyaXMxDjAMBgNVBAoMBXZpdGFtMRQwEgYDVQQLDAthdXRob3JpdGllczElMCMGA1UEAwwcY2FfaW50ZXJ
tZWRpYXRlX3RpbWVzdGFtcGluZwICANQwDQYJYIZIAWUDBAIDBQCgggEgMBoGCSqGSIb3DQEJAzENBgsqhkiG9w0
BCRABBDAcBgkqhkiG9w0BCQUxDxcNMTkxMTA3MDQwMDA5WjAtBgkqhkiG9w0BCTQxIDAeMA0GCWCGSAFlAwQCAwU
AoQ0GCSqGSIb3DQEBDQUAME8GCSqGSIb3DQEJBDFCBECaBL12eV9W7T0Y2Qbm43JWJDlmeXFeRgiwCTSfh91uIPd
xavc8huFSy5UuP9xeJ8gX7/NQ5ftuNy4SHxRM60YyMGQGCyqGSIb3DQEJEAIvMVUwUzBRME8

wCwYJYIZIAWUDBAIDBEAeduVLSR3MT88jYzzAMR8nbaiHoZPSUfMPxK/1Ekh3DXK7z46X0bKIs87kWN8Rjacz7Qu

F7nDQmPRcETZHjUjoMA0GCSqGSIb3DQEBDQUABIICALZUqEjXV7rYVWAgJ31YxJHADF3mYOv3aehjQlGMIuCzzmd
zgW9V3FagQvQB60rJg/sApdwgvRM4uAX9TG0sVKigsZ/sxhnhEEg5CEdP93R16y47/
TZYXs37Vqa3Qycd5xiEvL4zmJUl2ZA2SKptweiqt3VmWID4cZkYonTgwTgFIwev3YpDXd/
1Dy1zV5AnAs65bO37jVqxjVRUA2PUwxAvvOxG6uQLXj7GlbncTaQpSYhMErR1H0lpNvQ1F3D3O/
RpVSWnVjbToLTeADDsk4fJjsIU1fZeC3Im/
OZw1rKVzEDZdtuRvCr7LIst1JtxEcZTHDgiKmQXN4t39h+S49nBOldFq/
V43IdqMT+bjKxQlOG0x4dtnIvfDnH0eu93KuHr9ivGDp3GnDBLzePTX99kzM2Ky1vBeVL2+MzN27CIdceuZlQ1zp
ahXDgi+iOERhIf1GS+P2iR4vF4GKbUvsYQUrB2WNNaG/
naffGFDPVcd4tB6rJDOBR1TzgwaSEwUamlVebSkZBgvDIalzB12CuefhY+bMUPGbtQlWrrr6HizPTL2/
KrPakCoZk5nw9/
MICO+20eU6gP+s69TsV32CCTIpKxK87QD0NHz2ZBj5Ejte4h1sYxtR+3HvQYITaHlYxHMvy0KZVytMqCMVM/
Fyf2LGN9XjPd+ddDvyA7htRT",

      "destinationComparable" :
"MIILITAVAgEAMBAMDk9wZXJhdGlvbiBPa2F5MIILBgYJKoZIhvcNAQcCoIIK9zCCCvMCAQMxDzANBglghkgBZQM
EAgMFADCBgAYLKoZIhvcNAQkQAQSgcQRvMG0CAQEGASkwUTANBglghkgBZQMEAgMFAARACPaKQZakyyhRDnsAhDW
afWC2tL1A1Wg0AZIk2ycySfEFppfp1hY6Zpy0c82fZ7srdKPl+uWyqxH6a5q5lDbl5AIBARgPMjAxOTExMDcwNDA
wMDlaoIIGhzCCBoMwggRroAMCAQICAgDUMA0GCSqGSIb3DQEBCwUAMHgxCzAJBgNVBAYTAmZyMQwwCgYDVQQIDAN
pZGYxDjAMBgNVBAcMBXBhcmlzMQ4wDAYDVQQKDAV2aXRhbTEUMBIGA1UECwwLYXV0aG9yaXRpZXMxJTAjBgNVBAM
MHGNhX2ludGVybWVkaWF0ZV90aW1lc3RhbXBpbmcwHhcNMTkxMTA3MDIzMzA1WhcNMjIxMTA2MDIzMzA1WjBUMQs
wCQYDVQQGEwJmcjEMMAoGA1UECAwDaWRmMQ4wDAYDVQQHDAVwYXJpczEOMAwGA1UECgwFdml0YW0xFzAVBgNVBAM
MDnNlY3VyZS1sb2dib29rMIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEA09lqSj5nvYlemZC4CHCzTpH
NgK1WzA/
ILRjVL4ho2vfT6yx9TstAQqsVARrsknmAGJ2KMillffIOEZ00ph3H0oo7auWOuGDytsfteRMdFSVOxUfM/
1jBUXgqhg6VSHVka1uEAu0kg4g0ADvWrJbtVKHxps9vrEA+H4tEiTT3mfKEdWBAyLz4cGOln4ip3s1GMSjmVgDwV
gbzPnHvD1VUDLij6dYHbkAQaLSRh9hl47dPVbzOe1qZAUPDML6kVUI6fM9jxAzQJqub0jL+vEGvMUmJTqSTnFXAU
C5GOf3LU5iV/cwyGUWY0rz5kKS1XLT9jvYteijxYe4nf0y/V5Yzp/6F/
52esT4rqI5x9QGfv9TxSHiymcu316au14uFw+M+qLy4khC1BhNbsHojjnGhR33KuRa/
QAZltPJ3KyFC3NLSfGrJSi+JbZV5yS5yybkd1F25TdyUSFxk+0lQEvmL6ngsjKQHqSNdsTWmAHdBuE614ErrySxG
wjnFBMOfsPK2HKe3dc7iZUZYhPPUyBq8RVWDvHVFtQuXBCkCMSVNK6rVjmYMuSiI8oAy0TK7ixfAqMmE2yZwoJt7
```

```
hMYbo0Adrfjg7Bf95uP2LDC8LkWsALhIJdFOJT5ANXsWxGM0LmFmGTRoPFOb2E0G/
QQKAc0GEgRiG4qGihCLiZSGJ81CYw0CAwEAAaOCATkwggE1MCUGCWCGSAGG+EIBDQQYFhZDZXJ0aWZpY2F0IFNlc
nZldXIgU1NMMB0GA1UdDgQWBBRpel2gJ6UBuGBr8LG5d7Sa8kfmazCBmwYDVR0jBIGTMIGQgBQHzKjBQYoL+UCSH
cYIsFk+rST006F0pHIwcDELMAkGA1UEBhMCZnIxDDAKBgNVBAgMA2lkZjEOMAwGA1UEBwwFcGFyaXMxDjAMBgNVB
AoMBXZpdGFtMRQwEgYDVQQLDAthdXRob3JpdGllczEdMBsGA1UEAwwUY2Ffcm9vdF90aW1lc3RhbXBpbmeCAgDTM
AkGA1UdEgQCMAAwDAYDVR0TAQH/BAIwADALBgNVHQ8EBAMCBsAwEQYJYIZIAYb4QgEBBAQDAgZAMBYGA1UdJQEB/
wQMMAoGCCsGAQUFBwMIMA0GCSqGSIb3DQEBCwUAA4ICAQAwjDb7PvGAmqgW62+mXQHHo6hh8ikJyyz6kT10jLaRV
U6a77EMcUqF5Cl86d+tCpvGL5cdr8Z5WqZPAy3DyOxBDsGxIY9FM0lcrQAr6mY6xyTB+TrQDfqZcaqL0m1I/
PAWiQNwjDUJCDGEhvaoOm046fJNKcXrPjb9cQ/
CEsWdOJ4ERBZnUV8nOyVjCPQeW2yrY1hZrtYrfpTGl7thy0INGBz/zySHl5ArF1lbD75Z3o9ABMtRTqPjJ5R/
FeEzOYXZdpdc5C7UqQwyZXcQpRQlYJvedBDrBqGQ5AZ9KMnQEmQEHoE1zyxWoxpo2I1C/G/X/
nvBIiqgVUgsxTdh/btOIQPvKS/PuvmuqUeCb1KKwS7txgY4g5387q8NuVMrhUc6iux3J1mcX7ismaAk/
1AZPojzbNNxFwW5N/
4+q2TM8Q+0q3lxO9wyiGQqoCuP76ZvO0REukpFgUsBf+Yot1qyTxFYxpwgW1hX9ed8VsWJPHUNOy97Cd8MOPfgj+
CefBemvMa8zQH958KWtkjLlYKAhQuXHM5E6CuDUpE98090+ulNp4g7lSSJBaLOCMZWplji0syFRIGuirNloDruhi
lX/NOuGvBk5AUtvVvsbfFm3gOCpx+ijUMLcaBd5e6J0+xyvK2v0a+/x25hPjuLp/
weXmiMGtANyzbgmvJrNkgUDGCA80wggPJAgEBMH4weDELMAkGA1UEBhMCZnIxDDAKBgNVBAgMA2lkZjEOMAwGA1U
EBwwFcGFyaXMxDjAMBgNVBAoMBXZpdGFtMRQwEgYDVQQLDAthdXRob3JpdGllczElMCMGA1UEAwwcY2FfaW50ZXJ
tZWRpYXRlX3RpbWVzdGFtcGluZwICANQwDQYJYIZIAWUDBAIDBQCgggEgMBoGCSqGSIb3DQEJAzENBgsqhkiG9w0
BCRABBDAcBgkqhkiG9w0BCQUxDxcNMTkxMTA3MDQwMDA5WjAtBgkqhkiG9w0BCTQxIDAeMA0GCWCGSAFlAwQCAwU
AoQ0GCSqGSIb3DQEBDQUAME8GCSqGSIb3DQEJBDFCBECaBL12eV9W7T0Y2Qbm43JWJDlmeXFeRgiwCTSfh91uIPd
xavc8huFSy5UuP9xeJ8gX7/
NQ5ftuNy4SHxRM60YyMGQGCyqGSIb3DQEJEAIvMVUwUzBRME8wCwYJYIZIAWUDBAIDBEAeduVLSR3MT88jYzzAMR
8nbaiHoZPSUfMPxK/
1Ekh3DXK7z46X0bKIs87kWN8Rjacz7QuF7nDQmPRcETZHjUjoMA0GCSqGSIb3DQEBDQUABIICALZUqEjXV7rYVWA
gJ31YxJHADF3mYOv3aehjQlGMIuCzzmdzgW9V3FagQvQB60rJg/sApdwgvRM4uAX9TG0sVKigsZ/
sxhnhEEg5CEdP93R16y47/
TZYXs37Vqa3Qycd5xiEvL4zmJUl2ZA2SKptweiqt3VmWID4cZkYonTgwTgFIwev3YpDXd/
1Dy1zV5AnAs65bO37jVqxjVRUA2PUwxAvvOxG6uQLXj7GlbncTaQpSYhMErR1H0lpNvQ1F3D3O/
RpVSWnVjbToLTeADDsk4fJjsIU1fZeC3Im/
OZw1rKVzEDZdtuRvCr7LIst1JtxEcZTHDgiKmQXN4t39h+S49nBOldFq/
V43IdqMT+bjKxQlOG0x4dtnIvfDnH0eu93KuHr9ivGDp3GnDBLzePTX99kzM2Ky1vBeVL2+MzN27CIdceuZlQ1zp
ahXDgi+iOERhIf1GS+P2iR4vF4GKbUvsYQUrB2WNNaG/
naffGFDPVcd4tB6rJDOBR1TzgwaSEwUamlVebSkZBgvDIalzB12CuefhY+bMUPGbtQlWrrr6HizPTL2/
KrPakCoZk5nw9/
MICO+20eU6gP+s69TsV32CCTIpKxK87QD0NHz2ZBj5Ejte4h1sYxtR+3HvQYITaHlYxHMvy0KZVytMqCMVM/
Fyf2LGN9XjPd+ddDvyA7htRT",

      "action" : "COMPARISON",

      "item" : "PREVIOUS_TIMESTAMP_OPERATION",

      "status" : "OK"

    }, {

      "name" : "EVENTS_OPERATION_DATABASE_TRACEABILITY_COMPARISON",

      "details" : "Comparing operation ID found in database with operation ID found in
traceability secured file.",

      "type" : "LOCAL_INTEGRITY",

      "source" : "DATABASE",

      "destination" : "TRACEABILITY_FILE",

      "sourceComparable" : "aeeaaaaaachemhquabwdcaloiqpuvhiaaaaq",

      "destinationComparable" : "aeeaaaaaachemhquabwdcaloiqpuvhiaaaaq",

      "action" : "COMPARISON",

      "item" : "EVENT_OPERATION",

      "status" : "OK"

    }, {

      "name" : "TIMESTAMP_OBJECT_GROUP_DATABASE_TRACEABILITY_VALIDATION",
```

```
      "details" : "Validating timestamp object group found in database 'AND' timestamp
object group found in traceability secured file.",

      "type" : "TIMESTAMP_CHECKING",

      "source" : "DATABASE",

      "destination" : "TRACEABILITY_FILE",

      "sourceComparable" :
"MIILITAVAgEAMBAMDk9wZXJhdGlvbiBPa2F5MIILBgYJKoZIhvcNAQcCoIIK9zCCCvMCAQMxDzANBglghkgBZQM
EAgMFADCBgAYLKoZIhvcNAQkQAQSgcQRvMG0CAQEGASkwUTANBglghkgBZQMEAgMFAARAihHQjJ81c0AFrgQw145
Wkds5E2849TF7x0QrgWS3Z/Q6l1p7d0HIYU/
N0rAIGzUdqfaBONIvAjk99UL+XVol8QIBARgPMjAxOTExMDcwNTM4NDBaoIIGhzCCBoMwggRroAMCAQICAgDUMA0
GCSqGSIb3DQEBCwUAMHgxCzAJBgNVBAYTAmZyMQwwCgYDVQQIDANpZGYxDjAMBgNVBAcMBXBhcmlzMQ4wDAYDVQQ
KDAV2aXRhbTEUMBIGA1UECwwLYXV0aG9yaXRpZXMxJTAjBgNVBAMMHGNhX2ludGVybWVkaWF0ZV90aW1lc3RhbXB
pbmcwHhcNMTkxMTA3MDIzMzA1WhcNMjIxMTA2MDIzMzA1WjBUMQswCQYDVQQGEwJmcjEMMAoGA1UECAwDaWRmMQ4
wDAYDVQQHDAVwYXJpczEOMAwGA1UECgwFdml0YW0xFzAVBgNVBAMMDnNlY3VyZS1sb2dib29rMIICIjANBgkqhki
G9w0BAQEFAAOCAg8AMIICCgKCAgEA09lqSj5nvYlemZC4CHCzTpHNgK1WzA/
ILRjVL4ho2vfT6yx9TstAQqsVARrsknmAGJ2KMillffIOEZ00ph3H0oo7auWOuGDytsfteRMdFSVOxUfM/
1jBUXgqhg6VSHVka1ueAu0kg4g0ADvWrJbtVKHxps9vrEA+H4tEiTT3mfKEdWBAyLz4cGOln4ip3s1GMSjmVgDwV
gbzPnHvD1VUDLij6dYHbkAQaLSRh9hl47dPVbzOe1qZAUPDML6kVUI6fM9jxAzQJqub0jL+vEGvMUmJTqSTnFXAU
C5GOf3LU5iV/cwyGUWY0rz5kKS1XLT9jvYteijxYe4nf0y/V5Yzp/6F/
52esT4rqI5x9QGfv9TxSHiymcu316au14uFw+M+qLy4khC1BhNbsHojjnGhR33KuRa/
QAZltPJ3KyFC3NLSfGrJSi+JbZV5yS5yybkd1F25TdyUSFxk+0lQEvmL6ngsjKQHqSNdsTWmAHdBuE614ErrySxG
wjnFBMOfsPK2HKe3dc7iZUZYhPPUyBq8RVWDvHVFtQuXBCkCMSVNK6rVjmYMuSiI8oAy0TK7ixfAqMmE2yZwoJt7
hMYbo0Adrfjg7Bf95uP2LDC8LkWsALhIJdFOJT5ANXsWxGM0LmFmGTRoPFOb2E0G/
QQKAc0GEgRiG4qGihCLiZSGJ81CYw0CAwEAAaOCATkwggE1MCUGCWCGSAGG+EIBDQQYFhZDZXJ0aWZpY2F0IFNlc
nZldXIgU1NMMB0GA1UdDgQWBBRpel2gJ6UBuGBr8LG5d7Sa8kfmazCBmwYDVR0jBIGTMIGQgBQHzKjBQYoL+UCSH
cYIsFk+rST006F0pHIwcDELMAkGA1UEBhMCZnIxDDAKBgNVBAgMA2lkZjEOMAwGA1UEBwwFcGFyaXMxDjAMBgNVB
AoMBXZpdGFtMRQwEgYDVQQLDAthdXRob3JpdGllczEdMBsGA1UEAwwUY2Ffcm9vdF90aW1lc3RhbXBpbmeCAgDTM
AkGA1UdEgQCMAAwDAYDVR0TAQH/BAIwADALBgNVHQ8EBAMCBsAwEQYJYIZIAYb4QgEBBAQDAgZAMBYGA1UdJQEB/
wQMMAoGCCsGAQUFBwMIMA0GCSqGSIb3DQEBCwUAA4ICAQAwjDb7PvGAmqgW62+mXQHHo6hh8ikJyyz6kT10jLaRV
Uou77ZMoUqF5GlS6d+tSpvGL5cdrs2Z5WqZPny3DyOxBBsGnIY9FM0herQAr6mY6xyTB+TrQDfqZeaqL0m1I/
PAWiQNwjDUJCDGEhvaoOm046fJNKcXrPjb9cQ/
CEsWdOJ4ERBZnUV8nOyVjCPQeW2yrY1hZrtYrfpTGl7thy0INGBz/zySHl5ArF1lbD75Z3o9ABMtRTqPjJ5R/
FeEzOYXZdpdc5C7UqQwyZXcQpRQlYJvedBDrBqGQ5AZ9KMnQEmQEHoE1zyxWoxpo2I1C/G/X/
nvBIiqgVUgsxTdh/btOIQPvKS/PuvmuqUeCb1KKwS7txgY4g5387q8NuVMrhUc6iux3J1mcX7ismaAk/
1AZPojzbNNxFwW5N/
4+q2TM8Q+0q3lxO9wyiGQqoCuP76ZvO0REukpFgUsBf+Yot1qyTxFYxpwgW1hX9ed8VsWJPHUNOy97Cd8MOPfgj+
CefBemvMa8zQH958KWtkjLlYKAhQuXHM5E6CuDUpE98090+ulNp4g7lSSJBaLOCMZWplji0syFRIGuirNloDruhi
lX/NOuGvBk5AUtvVvsbfFm3gOCpx+ijUMLcaBd5e6J0+xyvK2v0a+/x25hPjuLp/
weXmiMGtANyzbgmvJrNkgUDGCA80wggPJAgEBMH4weDELMAkGA1UEBhMCZnIxDDAKBgNVBAgMA2lkZjEOMAwGA1U
EBwwFcGFyaXMxDjAMBgNVBAoMBXZpdGFtMRQwEgYDVQQLDAthdXRob3JpdGllczElMCMGA1UEAwwcY2FfaW50ZXJ
tZWRpYXRlX3RpbWVzdGFtcGluZwICANQwDQYJYIZIAWUDBAIDBQCgggEgMBoGCSqGSIb3DQEJAzENBgsqhkiG9w0
BCRABBDAcBgkqhkiG9w0BCQUxDxcNMTkxMTA3MDUzODQwWjAtBgkqhkiG9w0BCTQxIDAeMA0GCWCGSAFlAwQCAwU
AoQ0GCSqGSIb3DQEBDQUAME8GCSqGSIb3DQEJBDFCBEA8CxI4X4tNwBuejeHjuy8XzJ9BntyIAJU9hqh+JySsyL5
qsU89ZOxMZhgTEfWEeVGMCvhPqiSdWn7dRZvzRNnfMGQGCyqGSIb3DQEJEAIvMVUwUzBRME8wCwYJYIZIAWUDBAI
DBEAeduVLSR3MT88jYzzAMR8nbaiHoZPSUfMPxK/
1Ekh3DXK7z46X0bKIs87kWN8Rjacz7QuF7nDQmPRcETZHjUjoMA0GCSqGSIb3DQEBDQUABIICAC73jsTyXRCDvxU
9EV/0U4CrLR/y04wNlF3m+/LAFuJNXANo7OVTjyj8zEyG+s2X4/
uyQ6pSTI67yTAECFT6CnUqtaX2tGgDJHEvv16qMNsrEwKNFb8PnjiEj1mZf5E715hTOP+L0B/
EFZw7wr2yX5YvAECYBf2o/row9lS1MK590uMbkMcM6Eu/
yU6nOwY58ehiO2tAeRR6cF9Io6EeVOqQNfbMUpCSEVAT3f6+CiGON30nhK3bCuX7w9TG287IKE57CWmQcwyPi5N9
A5WBBIEzITxOstEeu1lHaq3GE0yjoi/
vv4dQeDrtfwi7Uywv1irxpaRzvel8UANM9peZxJ4an0ev4QbY43XZ1FsntOTNBHnGGHsnZlHED6r+CX59kjL8uTO
4dCljvcrGyrDw4g5i90SINjxtjteZ6746y2lXGnuOrdOTPRkQy22BWdf63eP4m5IvkCT4vK1dZbOmPklhDMSLvKb
pfqsJ14e5o5a7osNj3XYR+HklmajZFzD9Q6OvFgMtLcY4Bpob7pDKqim4XBdb1AasDYESleAhI4X8Oivx5Mh0iD2
vfBrINbLAAFn6UphpjOaWtVZgeK5NDwpTe0E9Rop2fqEnS9Zak9zIwtI8/
bLXtVnnNxE3SX+h6MovQ8QW8cK8vvcSQQbt9IyxLvXAcUD8gjOUoVwzkzmm",

      "destinationComparable" :
"MIILITAVAgEAMBAMDk9wZXJhdGlvbiBPa2F5MIILBgYJKoZIhvcNAQcCoIIK9zCCCvMCAQMxDzANBglghkgBZQM
EAgMFADCBgAYLKoZIhvcNAQkQAQSgcQRvMG0CAQEGASkwUTANBglghkgBZQMEAgMFAARAihHQjJ81c0AFrgQw145

Wkds5E2849TF7x0QrgWS3Z/Q6l1p7d0HIYU/
N0rAIGzUdqfaBONIvAjk99UL+XVol8QIBARgPMjAxOTExMDcwNTM4NDBaoIIGhzCCBoMwggRroAMCAQICAgDUMA0
GCSqGSIb3DQEBCwUAMHgxCzAJBgNVBAYTAmZyMQwwCgYDVQQIDANpZGYxDjAMBgNVBAcMBXBhcmlzMQ4wDAYDVQQ
KDAV2aXRhbTEUMBIGA1UECwwLYXV0aG9yaXRpZXMxJTAjBgNVBAM
```

```
MHGNhX2ludGVybWVkaWF0ZV90aW1lc3RhbXBpbmcwHhcNMTkxMTA3MDIzMzA1WhcNMjIxMTA2MDIzMzA1WjBUMQs
wCQYDVQQGEwJmcjEMMAoGA1UECAwDaWRmMQ4wDAYDVQQHDAVwYXJpczEOMAwGA1UECgwFdml0YW0xFzAVBgNVBAM
MDnNlY3VyZS1sb2dib29rMIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEA09lqSj5nvYlemZC4CHCzTpH
NgK1WzA/
ILRjVL4ho2vfT6yx9TstAQqsVARrsknmAGJ2KMillffIOEZ00ph3H0oo7auWOuGDytsfteRMdFSVOxUfM/
1jBUXgqhg6VSHVka1ueAu0kg4g0ADvWrJbtVKHxps9vrEA+H4tEiTT3mfKEdWBAyLz4cGOln4ip3s1GMSjmVgDwV
gbzPnHvD1VUDLij6dYHbkAQaLSRh9hl47dPVbzOe1qZAUPDML6kVUI6fM9jxAzQJqub0jL+vEGvMUmJTqSTnFXAU
C5GOf3LU5iV/cwyGUWY0rz5kKS1XLT9jvYteijxYe4nf0y/V5Yzp/6F/
52esT4rqI5x9QGfv9TxSHiymcu316au14uFw+M+qLy4khC1BhNbsHojjnGhR33KuRa/
QAZltPJ3KyFC3NLSfGrJSi+JbZV5yS5yybkd1F25TdyUSFxk+0lQEvmL6ngsjKQHqSNdsTWmAHdBuE614ErrySxG
wjnFBMOfsPK2HKe3dc7iZUZYhPPUyBq8RVWDvHVFtQuXBCkCMSVNK6rVjmYMuSiI8oAy0TK7ixfAqMmE2yZwoJt7
hMYbo0Adrfjg7Bf95uP2LDC8LkWsALhIJdFOJT5ANXsWxGM0LmFmGTRoPFOb2E0G/
QQKAc0GEgRiG4qGihCLiZSGJ81CYw0CAwEAAaOCATkwggE1MCUGCWCGSAGG+EIBDQQYFhZDZXJ0aWZpY2F0IFNlc
nZldXIgU1NMMB0GA1UdDgQWBBRpel2gJ6UBuGBr8LG5d7Sa8kfmazCBmwYDVR0jBIGTMIGQgBQHzKjBQYoL+UCSH
cYIsFk+rST006F0pHIwcDELMAkGA1UEBhMCZnIxDDAKBgNVBAgMA2lkZjEOMAwGA1UEBwwFcGFyaXMxDjAMBgNVB
AoMBXZpdGFtMRQwEgYDVQQLDAthdXRob3JpdGllczEdMBsGA1UEAwwUY2Ffcm9vdF90aW1lc3RhbXBpbmeCAgDTM
AkGA1UdEgQCMAAwDAYDVR0TAQH/BAIwADALBgNVHQ8EBAMCBsAwEQYJYIZIAYb4QgEBBAQDAgZAMBYGA1UdJQEB/
wQMMAoGCCsGAQUFBwMIMA0GCSqGSIb3DQEBCwUAA4ICAQAwjDb7PvGAmqgW62+mXQHHo6hh8ikJyyz6kT10jLaRV
Uou77ZMoUqF5GlS6d+tSpvGL5cdrs2Z5WqZPny3DyOxBBsGnIY9FM0herQAr6mY6xyTB+TrQDfqZeaqL0m1I/
PAWiQNwjDUJCDGEhvaoOm046fJNKcXrPjb9cQ/
CEsWdOJ4ERBZnUV8nOyVjCPQeW2yrY1hZrtYrfpTGl7thy0INGBz/zySHl5ArF1lbD75Z3o9ABMtRTqPjJ5R/
FeEzOYXZdpdc5C7UqQwyZXcQpRQlYJvedBDrBqGQ5AZ9KMnQEmQEHoE1zyxWoxpo2I1C/G/X/
nvBIiqgVUgsxTdh/btOIQPvKS/PuvmuqUeCb1KKwS7txgY4g5387q8NuVMrhUc6iux3J1mcX7ismaAk/
1AZPojzbNNxFwW5N/
4+q2TM8Q+0q3lxO9wyiGQqoCuP76ZvO0REukpFgUsBf+Yot1qyTxFYxpwgW1hX9ed8VsWJPHUNOy97Cd8MOPfgj+
CefBemvMa8zQH958KWtkjLlYKAhQuXHM5E6CuDUpE98090+ulNp4g7lSSJBaLOCMZWplji0syFRIGuirNloDruhi
lX/NOuGvBk5AUtvVvsbfFm3gOCpx+ijUMLcaBd5e6J0+xyvK2v0a+/x25hPjuLp/
weXmiMGtANyzbgmvJrNkgUDGCA80wggPJAgEBMH4weDELMAkGA1UEBhMCZnIxDDAKBgNVBAgMA2lkZjEOMAwGA1U
EBwwFcGFyaXMxDjAMBgNVBAoMBXZpdGFtMRQwEgYDVQQLDAthdXRob3JpdGllczElMCMGA1UEAwwcY2FfaW50ZXJ
tZWRpYXRlX3RpbWVzdGFtcGluZwICANQwDQYJYIZIAWUDBAIDBQCgggEgMBoGCSqGSIb3DQEJAzENBgsqhkiG9w0
BCRABBDAcBgkqhkiG9w0BCQUxDxcNMTkxMTA3MDUzODQwWjAtBgkqhkiG9w0BCTQxIDAeMA0GCWCGSAFlAwQCAwU
AoQ0GCSqGSIb3DQEBDQUAME8GCSqGSIb3DQEJBDFCBEA8CxI4X4tNwBuejeHjuy8XzJ9BntyIAJU9hqh+JySsyL5
qsU89ZOxMZhgTEfWEeVGMCvhPqiSdWn7dRZvzRNnfMGQGCyqGSIb3DQEJEAIvMVUwUzBRME8wCwYJYIZIAWUDBAI
DBEAeduVLSR3MT88jYzzAMR8nbaiHoZPSUfMPxK/
1Ekh3DXK7z46X0bKIs87kWN8Rjacz7QuF7nDQmPRcETZHjUjoMA0GCSqGSIb3DQEBDQUABIICAC73jsTyXRCDvxU
9EV/0U4CrLR/y04wNlF3m+/LAFuJNXANo7OVTjyj8zEyG+s2X4/
uyQ6pSTI67yTAECFT6CnUqtaX2tGgDJHEvv16qMNsrEwKNFb8PnjiEj1mZf5E715hTOP+L0B/
EFZw7wr2yX5YvAECYBf2o/row9lS1MK590uMbkMcM6Eu/
yU6nOwY58ehiO2tAeRR6cF9Io6EeVOqQNfbMUpCSEVAT3f6+CiGON30nhK3bCuX7w9TG287IKE57CWmQcwyPi5N9
A5WBBIEzITxOstEeu1lHaq3GE0yjoi/
vv4dQeDrtfwi7Uywv1irxpaRzvel8UANM9peZxJ4an0ev4QbY43XZ1FsntOTNBHnGGHsnZlHED6r+CX59kjL8uTO
4dCljvcrGyrDw4g5i90SINjxtjteZ6746y2lXGnuOrdOTPRkQy22BWdf63eP4m5IvkCT4vK1dZbOmPklhDMSLvKb
pfqsJ14e5o5a7osNj3XYR+HklmajZFzD9Q6OvFgMtLcY4Bpob7pDKqim4XBdb1AasDYESleAhI4X8Oivx5Mh0iD2
vfBrINbLAAFn6UphpjOaWtVZgeK5NDwpTe0E9Rop2fqEnS9Zak9zIwtI8/
bLXtVnnNxE3SX+h6MovQ8QW8cK8vvcSQQbt9IyxLvXAcUD8gjOUoVwzkzmm",

      "action" : "VALIDATION",

      "item" : "TIMESTAMP_OBJECT_GROUP",

      "status" : "OK"

    }, {

      "name" : "TIMESTAMP_OBJECT_GROUP_DATABASE_TRACEABILITY_COMPARISON",

      "details" : "Comparing timestamp object group found in database with timestamp
object group found in secured file.",

      "type" : "TIMESTAMP_CHECKING",

      "source" : "DATABASE",

      "destination" : "TRACEABILITY_FILE",

      "sourceComparable" :
"MIILITAVAgEAMBAMDk9wZXJhdGlvbiBPa2F5MIILBgYJKoZIhvcNAQcCoIIK9zCCCvMCAQMxDzANBglghkgBZQM
EAgMFADCBgAYLKoZIhvcNAQkQAQSgcQRvMG0CAQEGASkwUTANBglghkgBZQMEAgMFAARAihHQjJ81c0AFrgQw145
Wkds5E2849TF7x0QrgWS3Z/Q6l1p7d0HIYU/
N0rAIGzUdqfaBONIvAjk99UL+XVol8QIBARgPMjAxOTExMDcwNTM4NDBaoIIGhzCCBoMwggRroAMCAQICAgDUMA0
```

GCSqGSIb3DQEBCwUAMHgxCzAJBgNVBAYTAmZyMQwwCgYDVQQIDANpZGYxDjAMBgNVBAcMBXBhcmlzMQ4wDAYDVQQ
KDAV2aXRhbTEUMBIGA1UECwwLYXV0aG9yaXRpZXMxJTAjBgNVBAMMHGNhX2ludGVybWVkaWF0ZV90aW1lc3RhbXB
pbmcwHhcNMTkxMTA3MDIzMzA1WhcNMjIxMTA2MDIzMzA1WjBUMQswCQYDVQQGEwJmcjEMMAoGA1UECAwDaWRmMQ4
wDAYDVQQHDAVwYXJpczEOMAwGA1UECgwFdml0YW0xFzAVBgNVBAMMDnNlY3VyZS1sb2dib29rMIICIjANBgkqhki
G9w0BAQEFAAOCAg8AMIICCgKCAgEA09lqSj5nvYlemZC4CHCzTpHNgK1WzA/
ILRjVL4ho2vfT6yx9TstAQqsVARrsknmAGJ2KMillffIOEZ00ph3H0oo7auWOuGDytsfteRMdFSVOxUfM/
1jBUXgqhg6VSHVka1ueAu0kg4g0ADvWrJbtVKHxps9vrEA+H4tEiTT3mfKEdWBAyLz4cGOln4ip3s1GMSjmVgD

wVgbzPnHvD1VUDLij6dYHbkAQaLSRh9hl47dPVbzOe1qZAUPDML6kVUI6fM9jxAzQJqub0jL+vEGvMUmJTqSTnFX
AUC5GOf3LU5iV/cwyGUWY0rz5kKS1XLT9jvYteijxYe4nf0y/V5Yzp/6F/
52esT4rqI5x9QGfv9TxSHiymcu316au14uFw+M+qLy4khC1BhNbsHojjnGhR33KuRa/
QAZltPJ3KyFC3NLSfGrJSi+JbZV5yS5yybkd1F25TdyUSFxk+0lQEvmL6ngsjKQHqSNdsTWmAHdBuE614ErrySxG
wjnFBMOfsPK2HKe3dc7iZUZYhPPUyBq8RVWDvHVFtQuXBCkCMSV

NK6rVjmYMuSiI8oAy0TK7ixfAqMmE2yZwoJt7hMYbo0Adrfjg7Bf95uP2LDC8LkWsALhIJdFOJT5ANXsWxGM0LmF
mGTRoPFOb2E0G/
QQKAc0GEgRiG4qGihCLiZSGJ81CYw0CAwEAAaOCATkwggE1MCUGCWCGSAGG+EIBDQQYFhZDZXJ0aWZpY2F0IFNlc
nZldXIgU1NMMB0GA1UdDgQWBBRpel2gJ6UBuGBr8LG5d7Sa8kfmazCBmwYDVR0jBIGTMIGQgBQHzKjBQYoL+UCSH
cYIsFk+rST006F0pHIwcDELMAkGA1UEBhMCZnIxDDAKBgNVBAgMA2lkZjEOMAwGA1UEBwwFcGFyaXMxDjAMBgNVB
AoMBXZpdGFtMRQwEgYDVQQLDAthdXRob3JpdGllczEdMBsGA1UEAwwUY2Ffcm9vdF90aW1lc3RhbXBpbmeCAgDTM
AkGA1UdEgQCMAAwDAYDVR0TAQH/BAIwADALBgNVHQ8EBAMCBsAwEQYJYIZIAYb4QgEBBAQDAgZAMBYGA1UdJQEB/
wQMMAoGCCsGAQUFBwMIMA0GCSqGSIb3DQEBCwUAA4ICAQAwjDb7PvGAmqgW62+mXQHHo6hh8ikJyyz6kT10jLaRV
Uou77ZMoUqF5GlS6d+tSpvGL5cdrs2Z5WqZPny3DyOxBBsGnIY9FM0herQAr6mY6xyTB+TrQDfqZeaqL0m1I/
PAWiQNwjDUJCDGEhvaoOm046fJNKcXrPjb9cQ/
CEsWdOJ4ERBZnUV8nOyVjCPQeW2yrY1hZrtYrfpTGl7thy0INGBz/zySHl5ArF1lbD75Z3o9ABMtRTqPjJ5R/
FeEzOYXZdpdc5C7UqQwyZXcQpRQlYJvedBDrBqGQ5AZ9KMnQEmQEHoE1zyxWoxpo2I1C/G/X/
nvBIiqgVUgsxTdh/btOIQPvKS/PuvmuqUeCb1KKwS7txgY4g5387q8NuVMrhUc6iux3J1mcX7ismaAk/
1AZPojzbNNxFwW5N/
4+q2TM8Q+0q3lxO9wyiGQqoCuP76ZvO0REukpFgUsBf+Yot1qyTxFYxpwgW1hX9ed8VsWJPHUNOy97Cd8MOPfgj+
CefBemvMa8zQH958KWtkjLlYKAhQuXHM5E6CuDUpE98090+ulNp4g7lSSJBaLOCMZWplji0syFRIGuirNloDruhi
lX/NOuGvBk5AUtvVvsbfFm3gOCpx+ijUMLcaBd5e6J0+xyvK2v0a+/x25hPjuLp/
weXmiMGtANyzbgmvJrNkgUDGCA80wggPJAgEBMH4weDELMAkGA1UEBhMCZnIxDDAKBgNVBAgMA2lkZjEOMAwGA1U
EBwwFcGFyaXMxDjAMBgNVBAoMBXZpdGFtMRQwEgYDVQQLDAthdXRob3JpdGllczElMCMGA1UEAwwcY2FfaW50ZXJ
tZWRpYXRlX3RpbWVzdGFtcGluZwICANQwDQYJYIZIAWUDBAIDBQCgggEgMBoGCSqGSIb3DQEJAzENBgsqhkiG9w0
BCRABBDAcBgkqhkiG9w0BCQUxDxcNMTkxMTA3MDUzODQwWjAtBgkqhkiG9w0BCTQxIDAeMA0GCWCGSAFlAwQCAwU
AoQ0GCSqGSIb3DQEBDQUAME8GCSqGSIb3DQEJBDFCBEA8CxI4X4tNwBuejeHjuy8XzJ9BntyIAJU9hqh+JySsyL5
qsU89ZOxMZhgTEfWEeVGMCvhPqiSdWn7dRZvzRNnfMGQGCyqGSIb3DQEJEAIvMVUwUzBRME8wCwYJYIZIAWUDBAI
DBEAeduVLSR3MT88jYzzAMR8nbaiHoZPSUfMPxK/
1Ekh3DXK7z46X0bKIs87kWN8Rjacz7QuF7nDQmPRcETZHjUjoMA0GCSqGSIb3DQEBDQUABIICAC73jsTyXRCDvxU
9EV/0U4CrLR/y04wNlF3m+/LAFuJNXANo7OVTjyj8zEyG+s2X4/
uyQ6pSTI67yTAECFT6CnUqtaX2tGgDJHEvv16qMNsrEwKNFb8PnjiEj1mZf5E715hTOP+L0B/
EFZw7wr2yX5YvAECYBf2o/row9lS1MK590uMbkMcM6Eu/
yU6nOwY58ehiO2tAeRR6cF9Io6EeVOqQNfbMUpCSEVAT3f6+CiGON30nhK3bCuX7w9TG287IKE57CWmQcwyPi5N9
A5WBBIEzITxOstEeu1lHaq3GE0yjoi/
vv4dQeDrtfwi7Uywv1irxpaRzvel8UANM9peZxJ4an0ev4QbY43XZ1FsntOTNBHnGGHsnZlHED6r+CX59kjL8uTO
4dCljvcrGyrDw4g5i90SINjxtjteZ6746y2lXGnuOrdOTPRkQy22BWdf63eP4m5IvkCT4vK1dZbOmPklhDMSLvKb
pfqsJ14e5o5a7osNj3XYR+HklmajZFzD9Q6OvFgMtLcY4Bpob7pDKqim4XBdb1AasDYESleAhI4X8Oivx5Mh0iD2
vfBrINbLAAFn6UphpjOaWtVZgeK5NDwpTe0E9Rop2fqEnS9Zak9zIwtI8/
bLXtVnnNxE3SX+h6MovQ8QW8cK8vvcSQQbt9IyxLvXAcUD8gjOUoVwzkzmm",

"destinationComparable" :
"MIILITAVAgEAMBAMDk9wZXJhdGlvbiBPa2F5MIILBgYJKoZIhvcNAQcCoIIK9zCCCvMCAQMxDzANBglghkgBZQM
EAgMFADCBgAYLKoZIhvcNAQkQAQSgcQRvMG0CAQEGASkwUTANBglghkgBZQMEAgMFAARAihHQjJ81c0AFrgQw145
Wkds5E2849TF7x0QrgWS3Z/Q6l1p7d0HIYU/
N0rAIGzUdqfaBONIvAjk99UL+XVol8QIBARgPMjAxOTExMDcwNTM4NDBaoIIGhzCCBoMwggRroAMCAQICAgDUMA0
GCSqGSIb3DQEBCwUAMHgxCzAJBgNVBAYTAmZyMQwwCgYDVQQIDANpZGYxDjAMBgNVBAcMBXBhcmlzMQ4wDAYDVQQ
KDAV2aXRhbTEUMBIGA1UECwwLYXV0aG9yaXRpZXMxJTAjBgNVBAMMHGNhX2ludGVybWVkaWF0ZV90aW1lc3RhbXB
pbmcwHhcNMTkxMTA3MDIzMzA1WhcNMjIxMTA2MDIzMzA1WjBUMQswCQYDVQQGEwJmcjEMMAoGA1UECAwDaWRmMQ4
wDAYDVQQHDAVwYXJpczEOMAwGA1UECgwFdml0YW0xFzAVBgNVBAMMDnNlY3VyZS1sb2dib29rMIICIjANBgkqhki
G9w0BAQEFAAOCAg8AMIICCgKCAgEA09lqSj5nvYlemZC4CHCzTpHNgK1WzA/
ILRjVL4ho2vfT6yx9TstAQqsVARrsknmAGJ2KMillffIOEZ00ph3H0oo7auWOuGDytsfteRMdFSVOxUfM/
1jBUXgqhg6VSHVka1ueAu0kg4g0ADvWrJbtVKHxps9vrEA+H4tEiTT3mfKEdWBAyLz4cGOln4ip3s1GMSjmVgDwV
gbzPnHvD1VUDLij6dYHbkAQaLSRh9hl47dPVbzOe1qZAUPDML6kVUI6fM9jxAzQJqub0jL+vEGvMUmJTqSTnFXAU
C5GOf3LU5iV/cwyGUWY0rz5kKS1XLT9jvYteijxYe4nf0y/V5Yzp/6F/
52esT4rqI5x9QGfv9TxSHiymcu316au14uFw+M+qLy4khC1BhNbsHojjnGhR33KuRa/
QAZltPJ3KyFC3NLSfGrJSi+JbZV5yS5yybkd1F25TdyUSFxk+0lQEvmL6ngsjKQHqSNdsTWmAHdBuE614ErrySxG
wjnFBMOfsPK2HKe3dc7iZUZYhPPUyBq8RVWDvHVFtQuXBCkCMSVNK6rVjmYMuSiI8oAy0TK7ixfAqMmE2yZwoJt7

```
hMYbo0Adrfjg7Bf95uP2LDC8LkWsALhIJdFOJT5ANXsWxGM0LmFmGTRoPFOb2E0G/
QQKAc0GEgRiG4qGihCLiZSGJ81CYw0CAwEAAaOCATkwggE1MCUGCWCGSAGG+EIBDQQYFhZDZXJ0aWZpY2F0IFNlc
nZldXIgU1NMMB0GA1UdDgQWBBRpel2gJ6UBuGBr8LG5d7Sa8kfmazCBmwYDVR0jBIGTMIGQgBQHzKjBQYoL+UCSH
cYIsFk+rST006F0pHIwcDELMAkGA1UEBhMCZnIxDDAKBgNVBAgMA2lkZjEOMAwGA1UEBwwFcGFyaXMxDjAMBgNVB
AoMBXZpdGFtMRQwEgYDVQQLDAthdXRob3JpdGllczEdMBsGA1UEAwwUY2Ffcm9vdF90aW1lc3RhbXBpbmeCAgDTM
AkGA1UdEgQCMAAwDAYDVR0TAQH/BAIwADALBgNVHQ8EBAMCBsAwEQYJYIZIAYb4QgEBBAQDAgZAMBYGA1UdJQEB/
wQMMAoGCCsGAQUFBwMIMA0GCSqGSIb3DQEBCwUAA4ICAQAwjDb7PvGAmqgW62+mXQHHo6hh8ikJyyz6kT10jLaRV
Uou77ZMoUqF5GlS6d+tSpvGL5cdrs2Z5WqZPny3DyOxBBsGnIY9FM0herQAr6mY6xyTB+TrQDfqZeaqL0m1I/
PAW1QNwjDUJCDGEhvaoOm046fJNKcXrPjb9cQ/
CEsWdOJ4ERBZnUV8nOyVjCPQeW2yrY1hZrtYrfpTGl7thy0INGBz/zySHl5ArF1lbD75Z3o9ABMtRTqPjJ5R/
FeEzOYXZdpdc5C7UqQwyZXcQpRQlYJvedBDrBqGQ5AZ9KMnQEmQEHoE1zyxWoxpo2I1C/G/X/
nvBIiqgVUgsxTdh/btOIQPvKS/PuvmuqUeCb1KKwS7txgY4g5387q8NuVMrhUc6iux3J1mcX7ismaAk/
1AZPojzbNNxFwW5N/
4+q2TM8Q+0q3lxO9wyiGQqoCuP76ZvO0REukpFgUsBf+Yot1qyTxFYxpwgW1hX9ed8VsWJPHUNOy97Cd8MOPfgj+
CefBemvMa8zQH958KWtkjLlYKAhQuXHM5E6CuDUpE98090+ulNp4g7lSSJBaLOCMZWplji0syFRIGuirNloDruhi
lX/NOuGvBk5AUtvVvsbfFm3gOCpx+ijUMLcaBd5e6J0+xyvK2v0a+/x25hPjuLp/
weXmiMGtANyzbgmvJrNkgUDGCA80wggPJAgEBMH4weDELMAkGA1UEBhMCZnIxDDAKBgNVBAgMA2lkZjEOMAwGA1U
EBwwFcGFyaXMxDjAMBgNVBAoMBXZpdGFtMRQwEgYDVQQLDAthdXRob3JpdGllczElMCMGA1UEAwwcY2FfaW50ZXJ
tZWRpYXRlX3RpbWVzdGFtcGluZwICANQwDQYJYIZIAWUDBAIDBQCgggEgMBoGCSqGSIb3DQE

JAzENBgsqhkiG9w0BCRABBDAcBgkqhkiG9w0BCQUxDxcNMTkxMTA3MDUzODQwWjAtBgkqhkiG9w0BCTQxIDAeMA0

GCWCGSAFlAwQCAwUAoQ0GCSqGSIb3DQEBDQUAME8GCSqGSIb3DQEJBDFCBEA8CxI4X4tNwBuejeHjuy8XzJ9Bnty
IAJU9hqh+JySsyL5qsU89ZOxMZhgTEfWEeVGMCvhPqiSdWn7dRZvzRNnfMGQGCyqGSIb3DQEJEAIvMVUwUzBRME8
wCwYJYIZIAWUDBAIDBEAeduVLSR3MT88jYzzAMR8nbaiHoZPSUfMPxK/
1Ekh3DXK7z46X0bKIs87kWN8Rjacz7QuF7nDQmPRcETZHjUjoMA0GCSqGSIb3DQEBDQUABIICAC73jsTyXRCDvxU
9EV/0U4CrLR/y04wNlF3m+/LAFuJNXAN

7OVTjyj8zEyG+s2X4/
uyQ6pSTI67yTAECFT6CnUqtaX2tGgDJHEvv16qMNsrEwKNFb8PnjiEj1mZf5E715hTOP+L0B/
EFZw7wr2yX5YvAECYBf2o/row9lS1MK590uMbkMcM6Eu/
yU6nOwY58ehiO2tAeRR6cF9Io6EeVOqQNfbMUpCSEVAT3f6+CiGON30nhK3bCuX7w9TG287IKE57CWmQcwyPi5N9
A5WBBIEzITxOstEeu1lHaq3GE0yjoi/
vv4dQeDrtfwi7Uywv1irxpaRzvel8UANM9peZxJ4an0ev4QbY43XZ1FsntOTNBHnGGHsnZlHED6r+CX59kjL8uTO
4dCljvcrGyrDw4g5i90SINjxtjteZ6746y2lXGnuOrdOTPRkQy22BWdf63eP4m5IvkCT4vK1dZbOmPklhDMSLvKb
pfqsJ14e5o5a7osNj3XYR+HklmajZFzD9Q6OvFgMtLcY4Bpob7pDKqim4XBdb1AasDYESleAhI4X8Oivx5Mh0iD2
vfBrINbLAAFn6UphpjOaWtVZgeK5NDwpTe0E9Rop2fqEnS9Zak9zIwtI8/
bLXtVnnNxE3SX+h6MovQ8QW8cK8vvcSQQbt9IyxLvXAcUD8gjOUoVwzkzmm",

      "action" : "COMPARISON",

      "item" : "TIMESTAMP_OBJECT_GROUP",

      "status" : "OK"

    }, {

      "name" : "MERKLE_OBJECT_GROUP_DIGEST_DATABASE_TRACEABILITY_COMPARISON",

      "details" : "Comparing object group merkle digest found in database with object
group merkle digest found in traceability secured file.",

      "type" : "MERKLE_INTEGRITY",

      "source" : "DATABASE",

      "destination" : "TRACEABILITY_FILE",

      "sourceComparable" :
"CK8TOSHFhaw20w9veL19mg/6If9ogkJayMrWP11yY9u157YseSO97wnpJA3MaW6eoV2Mn9LjO6LxY86Gv1Roiw=
=",

      "destinationComparable" :
"CK8TOSHFhaw20w9veL19mg/6If9ogkJayMrWP11yY9u157YseSO97wnpJA3MaW6eoV2Mn9LjO6LxY86Gv1Roiw=
=",

      "action" : "COMPARISON",

      "item" : "MERKLE_TREE_ROOT_OBJECT_GROUP_DIGEST",

      "status" : "OK"
```

```
    }, {
      "name" : "MERKLE_OBJECT_GROUP_DIGEST_COMPUTATION_TRACEABILITY_COMPARISON",
      "details" : "Comparing object group merkle digest computed from secured data with
object group merkle digest found in traceability secured file.",
      "type" : "MERKLE_INTEGRITY",
      "source" : "COMPUTATION",
      "destination" : "TRACEABILITY_FILE",
      "sourceComparable" :
"CK8TOSHFhaw20w9veL19mg/6If9ogkJayMrWP11yY9u157YseSO97wnpJA3MaW6eoV2Mn9LjO6LxY86Gv1Roiw=
=",
      "destinationComparable" :
"CK8TOSHFhaw20w9veL19mg/6If9ogkJayMrWP11yY9u157YseSO97wnpJA3MaW6eoV2Mn9LjO6LxY86Gv1Roiw=
=",
      "action" : "COMPARISON",
      "item" : "MERKLE_TREE_ROOT_OBJECT_GROUP_DIGEST",
      "status" : "OK"
    }, {
      "name" :
"MERKLE_OBJECT_GROUP_DIGEST_COMPUTATION_ADDITIONAL_TRACEABILITY_COMPARISON",
      "details" : "Comparing object group merkle digest computed from secured data with
object group merkle digest found in additional traceability secured file.",
      "type" : "MERKLE_INTEGRITY",
      "source" : "COMPUTATION",
      "destination" : "ADDITIONAL_TRACEABILITY",
      "sourceComparable" :
"CK8TOSHFhaw20w9veL19mg/6If9ogkJayMrWP11yY9u157YseSO97wnpJA3MaW6eoV2Mn9LjO6LxY86Gv1Roiw=
=",
      "destinationComparable" :
"CK8TOSHFhaw20w9veL19mg/6If9ogkJayMrWP11yY9u157YseSO97wnpJA3MaW6eoV2Mn9LjO6LxY86Gv1Roiw=
=",
      "action" : "COMPARISON",
      "item" : "MERKLE_TREE_ROOT_OBJECT_GROUP_DIGEST",
      "status" : "OK"
    }, {
      "name" : "TIMESTAMP_OBJECT_GROUP_COMPUTATION_TRACEABILITY_COMPARISON",
      "details" : "Comparing timestamp object group computed from computing information
traceability file with timestamp object group found in secured file.",
      "type" : "TIMESTAMP_CHECKING",
      "source" : "COMPUTATION",
      "destination" : "TRACEABILITY_FILE",
      "sourceComparable" :
"ihHQjJ81c0AFrgQw145Wkds5E2849TF7x0QrgWS3Z/Q6l1p7d0HIYU/N0rAIGzUdqfaBONIvAjk99UL+XVol8Q=
=",
      "destinationComparable" : "ihHQjJ81c0AFrgQw145Wkds5E2849TF7x0QrgWS3Z/Q6l1p7d0HIYU/
```

```
N0rAIGzUdqfaBONIvAjk99UL+XVol8Q==",

      "action" : "COMPARISON",

      "item" : "TIMESTAMP_OBJECT_GROUP",

      "status" : "OK"

    }, {

      "name" : "PREVIOUS_TIMESTAMP_OBJECT_GROUP_DATABASE_TRACEABILITY_VALIDATION",

      "details" : "Validating previous timestamp object group found in database 'AND'
previous timestamp object group found in traceability secured file.",

      "type" : "CHAIN",

      "source" : "DATABASE",

      "destination" : "TRACEABILITY_FILE",

      "sourceComparable" :
"ihHQjJ81c0AFrgQw145Wkds5E2849TF7x0QrgWS3Z/Q6lFp7d0HIYU/N0rAIGzUdqfaBONIvAjk99UL+XVol8F=
=",

      "destinationComparable" : "ihHQjJ81c0AFrgQw145Wkds5E2849TF7x0QrgWS3Z/Q6lFp7d0HIYU/
N0rAIGzUdqfaBONIvAjk99UL+XVol8F==",

      "action" : "VALIDATION",

      "item" : "PREVIOUS_TIMESTAMP_OBJECT_GROUP",

      "status" : "OK"

    }, {

      "name" : "PREVIOUS_TIMESTAMP_OBJECT_GROUP_DATABASE_TRACEABILITY_COMPARISON",

      "details" : "Comparing previous timestamp object group found in database with
previous timestamp object group found in traceability secured file.",

      "type" : "CHAIN",

      "source" : "DATABASE",

      "destination" : "TRACEABILITY_FILE",

      "sourceComparable" :
"ihHQjJ81c0AFrgQw145Wkds5E2849TF7x0QrgWS3Z/Q6lFp7d0HIYU/N0rAIGzUdqfaBONIvAjk99UL+XVol8Q=
=",

      "destinationComparable" :

"ihHQjJ81c0AFrgQw145Wkds5E2849TF7x0QrgWS3Z/Q6lFp7d0HIYU/
N0rAIGzUdqfaBONIvAjk99UL+XVol8Q==",

      "action" : "COMPARISON",

      "item" : "PREVIOUS_TIMESTAMP_OBJECT_GROUP",

      "status" : "OK"

    }, {

      "name" : "FILE_DIGEST_DATABASE_TRACEABILITY_COMPARISON",

      "details" : "Comparing binary file digest found in object group database with
binary file digest found in traceability secured file.",

      "type" : "LOCAL_INTEGRITY",

      "source" : "DATABASE",
```

```
      "destination" : "TRACEABILITY_FILE",

      "sourceComparable" :
"9fe79f5153bd0b6996a23de06544d2af966dcc9e39cc46b49ae05bf6429bb88ff8d34b9bd0e6fd23a1b0681
136a9767976b6fb65cea41393e54a167e0752d620",

      "destinationComparable" :
"9fe79f5153bd0b6996a23de06544d2af966dcc9e39cc46b49ae05bf6429bb88ff8d34b9bd0e6fd23a1b0681
136a9767976b6fb65cea41393e54a167e0752d620",

      "action" : "COMPARISON",

      "item" : "FILE_DIGEST",

      "status" : "OK"

    }, {

      "name" : "EVENTS_OBJECT_GROUP_DIGEST_DATABASE_TRACEABILITY_COMPARISON",

      "details" : "Comparing object group lfc digest computed from logbook lifecycle
object group database with object group lfc digest found in traceability secured file.",

      "type" : "LOCAL_INTEGRITY",

      "source" : "DATABASE",

      "destination" : "TRACEABILITY_FILE",

      "sourceComparable" :
"UqMkFb5S2HCa/DJ92kadPX4ertC0IssWStvpA18UKEwYyYekTzFAeqIJPETGDeh6VBHLJsXWyvAGfwnqA36qCw=
=",

      "destinationComparable" :
"UqMkFb5S2HCa/DJ92kadPX4ertC0IssWStvpA18UKEwYyYekTzFAeqIJPETGDeh6VBHLJsXWyvAGfwnqA36qCw=
=",

      "action" : "COMPARISON",

      "item" : "EVENT_OBJECT_GROUP",

      "status" : "OK"

    }, {

      "name" : "FILE_DIGEST_OFFER_DATABASE_COMPARISON",

      "details" : "Comparing binary file digest stored in offers with binary file digest
found in object group database.",

      "type" : "LOCAL_INTEGRITY",

      "source" : "OFFER",

      "destination" : "DATABASE",

      "sourceComparable" :
"9fe79f5153bd0b6996a23de06544d2af966dcc9e39cc46b49ae05bf6429bb88ff8d34b9bd0e6fd23a1b0681
136a9767976b6fb65cea41393e54a167e0752d620",

      "destinationComparable" :
"9fe79f5153bd0b6996a23de06544d2af966dcc9e39cc46b49ae05bf6429bb88ff8d34b9bd0e6fd23a1b0681
136a9767976b6fb65cea41393e54a167e0752d620",

      "action" : "COMPARISON",

      "item" : "FILE_DIGEST",

      "status" : "OK"

    }, {

      "name" : "FILE_DIGEST_LFC_DATABASE_COMPARISON",
```

```
      "details" : "Comparing binary file digest found in object group database with
binary file digest found in logbook lifecycle object group database.",



      "type" : "LOCAL_INTEGRITY",

      "source" : "DATABASE",

      "destination" : "DATABASE",

      "sourceComparable" :
"9fe79f5153bd0b6996a23de06544d2af966dcc9e39cc46b49ae05bf6429bb88ff8d34b9bd0e6fd23a1b0681
136a9767976b6fb65cea41393e54a167e0752d620",

      "destinationComparable" :
"9fe79f5153bd0b6996a23de06544d2af966dcc9e39cc46b49ae05bf6429bb88ff8d34b9bd0e6fd23a1b0681
136a9767976b6fb65cea41393e54a167e0752d620",

      "action" : "COMPARISON",

      "item" : "FILE_DIGEST",

      "status" : "OK"

    } ],

    "evStartDateTime" : "2019-11-07T09:18:05.178",

    "evEndDateTime" : "2019-11-07T09:18:06.640",

    "status" : "OK"

  } ],

  "ReportVersion" : 2

}
```

2.7.2.Détails du rapport

La première partie du rapport « operationSummary » fait état des informations essentielles sur l'opération :

- « tenant » : le tenant sur lequel a été lancée l'opération consistant à générer le relevé de valeur probant
- « evId » : l'identifiant unique (GUID) de l'opération ;
- « evType » : le type de l'opération, ici EXPORT_PROBATIVE_VALUE ;
- « outcome » : le code de résultat global de l'opération (OK, KO ou WARNING) ;
- « outDetail » : le code de résultat propre à l'opération ;
- « outMsg » : le message explicite de résultat ;
- « rightsStatementIdentifier » : le contrat d'accès utilisé pour lancer l'opération.

La deuxième partie « reportSummary » est constituée du résumé du rapport :

- « evStartDateTime » : la date de l'enregistrement du premier statut dans le rapport ;
- « evStartDateTime »: la date de l'enregistrement du dernier statut dans le rapport ;
- « reportType »: le type de rapport, ici PROBATIVE_VALUE
- « vitamResults » : la liste des valeurs de statut d'exécution sur chaque objet, ici OK, KO et WARNING, ainsi que le total

La quatrième et dernière partie « reportEntries » rend compte des vérifications faites sur chaque objet binaire concerné, avec, pour chacun, un objet json contenant :

- « unitIds » : le tableau des unités archivistiques sélectionnées par la requête et contenant cet objet binaire (dans le cas général un seul) ;
- « objectGroupId » : l'identifiant unique du groupe d'objets techniques contenant cet objet binaire (GUID du GOT)
- « objectId » : l'identifiant unique de l'objet binaire concerné (GUID d'objet)
- « usageVersion » : le couple « usage_version » de l'objet concerné (ex. BinaryMaster$_1$) ;
- « operations » : un tableau contenant les opérations à vérifier sur l'objet binaire concerné, à savoir :
  - l'opération (entrée/INGEST ou préservation/PRESERVATION) qui a conduit à la prise en charge de cet objet binaire dans le système,
  - l'opération de sécurisation de cette opération de création ;
  - l'opération de sécurisation du journal de cycle de vie au moment de la création.
- Pour chacune de ces opérations sont fournies le type de l'opération, la date de fin de l'opération, ainsi que des éléments de contexte spécifiques aux opérations d'entrée/INGEST, à savoir :
  - le contrat d'entrée « rightsStatementIdentifier »,
  - le contexte applicatif d'entrée « agIdApp » et
  - le jeton fourni par l'applicatif d'entrée « evIdAppSession »
- « checks » : le tableau de toutes les vérifications faites sur l'objet binaire concerné. Ces vérifications sont toutes décrites selon le même modèle qui permet de comparer les deux valeurs provenant de sources différentes :
  - « name » : le code de la vérification ;
  - « details » : le libellé explicite de la vérification ;

- « type » : le type de la vérification à savoir « TIMESTAMP_CHECKING », « MERKLE_INTEGRITY », « CHAIN » ou « LOCAL_INTEGRITY »
- « source » : l'origine de l'information à vérifier à savoir « DATABASE » (base de données Mongo), « TRACEABILITY_FILE » (fichier de sécurisation) ou « OFFER » (offres de stockage)
- « destination » : l'origine de l'information permettant d'assurer la vérification à savoir « DATABASE », « TRACEABILITY_FILE » ou « OFFER »
- « sourceComparable » : la valeur à comparer venant de la source ;
- « destinationComparable » : la valeur à comparer venant de la destination ;
- « action » : l'action faite lors de la vérification, en général « COMPARAISON » mais peut être aussi « VALIDATION » si, outre la comparaison il est possible de vérifier le contenu de la valeur, ce qui est le cas pour un tampon d'horodatage ;
- « item » : la partie d'information de l'objet qui est vérifiée ;
- « status » : le résultat de la vérification
- « SecuredOperationId » : identifiant de l'opération de sécurisation
- « SecureOperationIdForOpId » : sécurisation de l'opération de sécurisation

- « Checks » : tableau retraçant les contrôles effectués sur les logbooks en question, conformité à la sécurisation et arbre de Merkle
  - « Name » : nom du logbook concerné « checkLogbookSecureInfoForOpi »
  - « Status » : statut du contrôle
  - « Name » : « CheckObjectHash », vérification du hash de l'objet
  - « Status » : statut du contrôle
  - « Name » : « checkLfcStorageEvent », vérification de la sécurisation des événements à la date de la dernière sauvegarde
  - « Status » : statut du contrôle
- « Operations Reports » : tableaux rassemblant les différents journaux des opérations logbooks, liste des opérations et vérification des logbooks en question
  - « EvTypeProc » : type d'opération, toujours « TRACEABILITY »
  - « Id » : identifiant de l'opération
  - « OperationCheckStatus » : le résultat du check de l'opération
  - « Details » : message de vérification de l'opération en question par rapport à la sécurisation
  - Dans le cas d'une opération d'entrée
    - « EvIdAppSession » : mention du contrat d'entrée utilisé lors de l'opération d'entrée
    - « agIdApp » : mention du contexte applicatif utilisé pour réaliser l'opération d'entrée

Et finalement le statut final de la vérification de l'objet.

3. Export d'un DIP

Workflow d'export d'un DIP

Cette section décrit le processus (workflow) d'export, utilisé lors de l'export d'un Dissemination Information Package (DIP) dans la solution logicielle Vitam.

Le workflow d'export de DIP actuel mis en place dans la solution logicielle Vitam est défini dans l'unique fichier *"ExportUnitWorkflow.json"*. Ce fichier est disponible dans /sources/processing/processing-management/src/main/resources/workflows.

Toutes les étapes, tâches et traitements sont journalisés dans le journal des opérations.et décrivent le processus (clé et description de la clé associée dans le journal des opérations) tel qu'implémenté dans la version actuelle de la solution logicielle Vitam.

3.1.Processus de création du bordereau de mise à disposition (STP_CREATE_MANIFEST)

3.1.1.Vérification des seuils de limitation de traitement des unités archivistiques CHECK_DISTRIBUTION_THRESHOLD

- **Règle** : tâche consistant à vérifier les seuils de limitation de traitement des unités archivistiques
- **Type** : bloquant
- **Statuts** :
  - OK : la vérification des seuils de limitation de traitement des unités archivistiques a bien été effectuée (CHECK_DISTRIBUTION_THRESHOLD.OK = Succès de la vérification des seuils de limitation de traitement des unités archivistiques)
  - KO : une incohérence à été détectée entre le seuil et le nombre d'unités archivistiques à traiter. (CHECK_DISTRIBUTION_THRESHOLD.KO = Échec de la vérification des seuils de limitation de traitement des unités archivistiques)
  - FATAL : une erreur technique est survenue lors de la vérification des seuils de limitation de traitement des unités archivistiques (CHECK_DISTRIBUTION_THRESHOLD.FATAL = Erreur technique lors de la vérification des seuils de limitation de traitement des unités archivistiques)

3.1.2.Création du bordereau CREATE_MANIFEST (CreateManifest.java)

- **Règle** : tâche consistant à créer le bordereau contenant les unités archivistiques soumises au service d'export de DIP, ainsi que les groupes d'objets techniques et objets qui leur sont associés
- **Type** : bloquant
- **Statuts** :
  - OK : le bordereau contenant les descriptions des unités archivistiques, groupes d'objets techniques et objets-données a été créé avec succès (CREATE_MANIFEST.OK = Succès de la création du bordereau de mise à disposition)
  - KO : la création du bordereau contenant les descriptions des unités archivistiques, groupes d'objets techniques et objets-données a échouée, car des informations étaient manquantes, erronées ou inconnues (CREATE_MANIFEST.KO = Échec de la création du bordereau de mise à disposition)
  - FATAL : une erreur technique est survenue lors de la création du bordereau (CREATE_MANIFEST.FATAL = Erreur technique lors de la création du bordereau de mise à disposition)

3.2.Processus de déplacement des objets binaires vers l'espace de travail interne (STP_PUT_BINARY_ON_WORKSPACE)

Déplacement des objets binaires vers le workspace PUT_BINARY_ON_WORKSPACE

(PutBinaryOnWorkspace.java)

- **Règle** : tâche consistant à déplacer les objets binaires mentionnés dans le bordereau vers l'espace de travail interne (« workspace »)
- **Type** : bloquant
- **Statuts** :
  - OK : les objets binaires ont été déplacés vers le workspace avec succès (PUT_BINARY_ON_WORKSPACE.OK = Succès du déplacement des objets binaires de l'offre de stockage vers l'espace de travail interne
  - KO : le déplacement des objets binaires vers le workspace a échoué car un ou plusieurs de ces objets étaient introuvables (PUT_BINARY_ON_WORKSPACE.KO = Échec du déplacement des objets binaires de l'offre de stockage vers l'espace de travail interne)
  - FATAL : une erreur technique est survenue lors du déplacement des objets binaires de l'offre de stockage vers le workspace (PUT_BINARY_ON_WORKSPACE.FATAL = Erreur technique lors du déplacement des objets binaires de l'offre de stockage vers l'espace de travail interne)

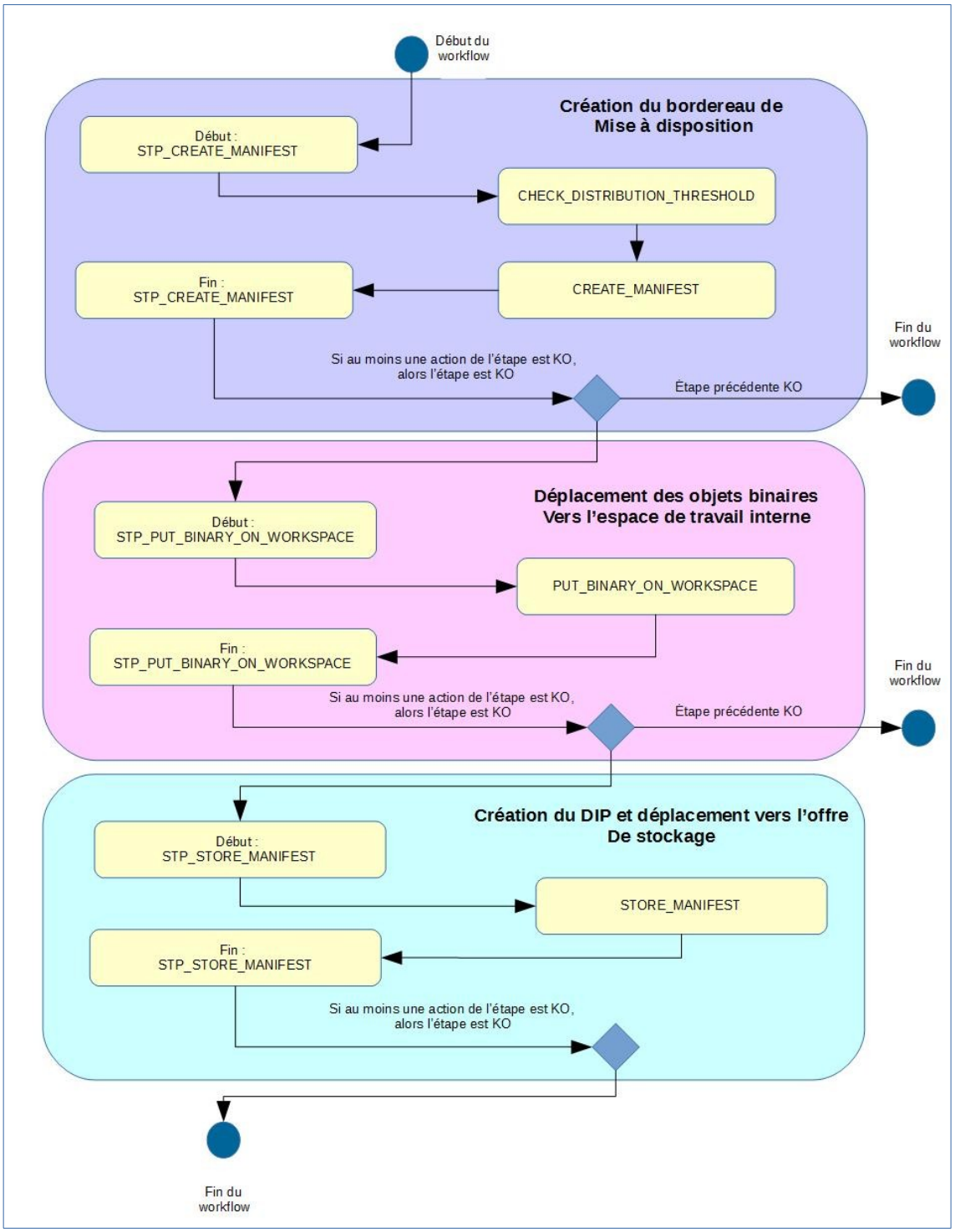
3.3.Processus de création du DIP et de son déplacement vers l'offre de stockage (STP_STORE_MANIFEST)

Stockage du DIP STORE_MANIFEST (StoreDIP.java)

- **Règle** : tâche consistant à créer le DIP et à le déplacer vers l'offre de stockage
- **Type** : bloquant
- **Statuts** :
  - OK : le DIP a été créé et enregistré sur les offres de stockage avec succès (STORE_MANIFEST.OK = Succès du processus de la création du DIP et de son déplacement vers l'offre de stockage)
  - FATAL : une erreur technique est survenue lors de la création et de l'enregistrement du DIP sur les offres de stockage déplacement des objets binaires de stockage vers l'espace de travail interne (STORE_MANIFEST.FATAL = Erreur technique lors de la création du DIP et de son déplacement vers l'offre de stockage)

3.4.Structure du Workflow d'export de DIP

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



4. INGEST

4.1.Workflow d'entrée

Cette section décrit le processus (workflow) d'entrée, utilisé lors du transfert d'un Submission Information Package (SIP) dans la solution logicielle Vitam. Ce workflow se décompose en deux grandes catégories : le processus d'entrée externe dit « ingest externe » et le processus d'entrée interne dit « ingest interne ». Le premier prend en charge le SIP et effectue des contrôles techniques préalables, tandis que le second débute dès le premier traitement métier. Ex: Le processus d'entrée externe comprend l'étape : STP_SANITY_CHECK_SIP (Contrôle sanitaire du SIP). Les autres étapes font partie du processus d'entrée interne.

Toutes les étapes, tâches et traitements sont journalisés dans le journal des opérations et décrivent le processus (clé et description de la clé associée dans le journal des opérations) tel qu'implémenté dans la version actuelle de la solution logicielle Vitam.

4.1.1.Processus des contrôles préalables à l'entrée (STP_SANITY_CHECK_SIP)

4.1.1.1. Contrôle sanitaire du SIP SANITY_CHECK_SIP (IngestExternalImpl.java)

- **Règle** : tâche consistant à vérifier l'absence de virus dans le SIP
- **Type** : bloquant
- **Statuts** :
  - OK : aucun virus n'a été détecté dans le SIP (SANITY_CHECK_SIP.OK = Succès du processus des contrôles préalables à l'entrée)
  - KO : un ou plusieurs virus ont été détectés dans le SIP (SANITY_CHECK_SIP.KO = Échec du processus des contrôles préalables à l'entrée)
  - FATAL : une erreur technique est survenue lors de la vérification de la présence de virus dans le SIP (SANITY_CHECK_SIP.FATAL = Erreur technique lors du processus des contrôles préalables à l'entrée)

4.1.1.2. Contrôle du format du conteneur du SIP CHECK_CONTAINER (IngestExternalImpl.java)

- **Règle** : tâche consistant à vérifier le format du SIP via un outil d'identification de format qui se base sur le référentiel des formats qu'il intègre

  Formats acceptés : .zip, .tar, .tar.gz, .tar.bz2 et tar.gz2
- **Type** : bloquant
- **Statuts** :
  - OK : le conteneur du SIP est au bon format (CHECK_CONTAINER.OK = Succès du contrôle du format du conteneur du SIP)
  - KO : le conteneur du SIP n'est pas au bon format (CHECK_CONTAINER.KO = Échec du contrôle du format du conteneur du SIP)
  - FATAL : une erreur technique est survenue lors de la vérification du format du conteneur du SIP, liée à l'outil d'identification des formats (CHECK_CONTAINER.FATAL = Erreur technique lors du contrôle du format du conteneur du SIP)

4.1.1.3. Contrôle du nom du bordereau de transfert MANIFEST_FILE_NAME_CHECK (IngestExternalImpl.java)

- **Règle** : tâche consistant à vérifier le nom du bordereau de transfert. Le nom du bordereau doit être conforme avec l'expression régulière suivante :

  ^(([a-zA-Z0-9]{1,56}[_-]{1}){0,1}|_{0,1})(manifest.xml)\b »} A savoir : une chaîne de caractères

débutant par des caractères alphanumériques sans accent, jusqu'à 56 caractères, suivi d'un « - » ou d'un « _ » puis suivi de « manifest.xml ». Exemples valides : « MonNouveau-manifest.xml », « UnAutreBordereau_manifest.xml ». « manifest.xml » tout court est également valide. Un SIP ne possédant pas du tout de bordereau de transfert verra également son entrée terminer en KO à cette tâche.

- **Type** : bloquant
- **Statuts** :
  - OK : le nom du bordereau de transfert est conforme (MANIFEST_FILE_NAME_CHECK.OK = Succès du contrôle du nom du bordereau de transfert : nom du fichier conforme)
  - KO : le nom du bordereau de transfert n'est pas conforme (MANIFEST_FILE_NAME_CHECK.KO = Échec du contrôle du nom du bordereau de transfert : nom du fichier non conforme)
  - FATAL : une erreur technique est survenue lors de la vérification du nom du bordereau de transfert (MANIFEST_FILE_NAME_CHECK.FATAL = Erreur technique lors du contrôle du nom du bordereau de transfert)

4.1.1.4. Vérification de l'intégrité du bordereau de transfert MANIFEST_DIGEST_CHECK (IngestExternalImpl.java)

- **Règle** : tâche consistant à vérifier l'empreinte du bordereau de transfert calculée par la solution logicielle Vitam et celle déclarée dans le bordereau. Si l'empreinte déclarée dans le bordereau de transfert n'a pas été calculée avec l'algorithme SHA-512, alors l'empreinte est recalculée avec cet algorithme. Elle sera alors enregistrée dans la solution logicielle Vitam.

  Algorithmes autorisés en entrée : MD5, SHA-1, SHA-256, SHA-512
- **Type** : bloquant
- **Statuts** :
  - KO : L'empreinte du bordereau de transfert n'est pas conforme (MANIFEST_DIGEST_CHECK.KO=Échec du contrôle de l''empreinte du bordereau de transfert : fichier non conforme)
  - FATAL : Une erreur technique est survenue lors du contrôle de l'empreinte (MANIFEST_DIGEST_CHECK.FATAL=Erreur technique lors du contrôle de l''empreinte du bordereau de transfert)

4.1.2. Processus de réception du SIP dans Vitam STP_UPLOAD_SIP (IngestInternalResource.java)

- **Règle** : étape consistant à vérifier la bonne réception du SIP sur l'espace de travail interne (« workspace »)
- **Type** : bloquant
- **Statuts** :
  - OK : le SIP a été réceptionné sur l'espace de travail interne (STP_UPLOAD_SIP.OK = Succès du processus de réception du SIP)
  - KO : le SIP n'a pas été réceptionné sur l'espace de travail interne (STP_UPLOAD_SIP.KO = Échec du processus de réception du SIP)
  - FATAL : une erreur technique est survenue lors de la réception du SIP dans la solution logicielle Vitam, par exemple suite à une indisponibilité du serveur (STP_UPLOAD_SIP.FATAL = Erreur technique lors du processus de réception du SIP)

4.1.3.Processus de contrôle du SIP (STP_INGEST_CONTROL_SIP)

4.1.3.1. Vérification globale du CHECK_SEDA (CheckSedaActionHandler.java)

- **Règle** : tâche consistant à vérifier la cohérence physique du SIP reçu par rapport au modèle de SIP accepté

  Type de SIP accepté : le bordereau de transfert, obligatoire dans le SIP, doit être conforme au schéma xsd par défaut fourni avec le standard SEDA v. 2.1, le SIP doit satisfaire les exigences du document « Structuration des SIP » et doit posséder un répertoire unique nommé « Content ».
- **Type** : bloquant
- **Statuts** :
  - OK : le SIP est présent et conforme au schéma xsd par défaut fourni avec le standard SEDA v.2.1. il satisfait aux exigences de « structuration des SIP » et possède un repertoire unique « Content » (CHECK_SEDA.OK = Succès de la vérification globale du SIP)
  - KO :
    - Cas 1 : le bordereau de transfert est introuvable dans le SIP ou n'est pas au format XML (CHECK_SEDA.NO_FILE.KO = Absence du bordereau de transfert ou bordereau de transfert au mauvais format)
    - Cas 2 : le bordereau de transfert n'est pas au format XML (CHECK_SEDA.NOT_XML_FILE.KO = Échec de la vérification globale du SIP : bordereau de transfert non conforme aux caractéristiques d'un fichier xml)
    - Cas 3 : le bordereau de transfert ne respecte pas le schéma par défaut fourni avec le standard SEDA 2.1 (CHECK_SEDA.NOT_XSD_VALID.KO = Échec de la vérification globale du SIP : bordereau de transfert non conforme au schéma SEDA 2.1)
    - Cas 4 : le SIP contient plus d'un dossier « Content » (CHECK_SEDA.CONTAINER_FORMAT.DIRECTORY.KO = Échec de la vérification globale du SIP : le SIP contient plus d'un dossier ou un dossier dont le nommage est invalide)
    - Cas 5 : le SIP contient plus d'un seul fichier à la racine (CHECK_SEDA.CONTAINER_FORMAT.FILE.KO = Échec de la vérification globale du SIP : le SIP contient plus d'un fichier à sa racine)
    - Cas 6 : l'action est déjà exécutée CHECK_SEDA.ALREADY_EXECUTED = Action déjà exécutée : Pas de vérification globale du SIP
  - FATAL :
    - Cas 1 : une erreur technique est survenue lors de la vérification globale du SIP (CHECK_SEDA.FATAL = Erreur technique lors de la vérification globale du SIP)
    - Cas 2 : une erreur technique est survenue lors de la vérification globale du SIP (CHECK_SEDA.NOT_XML_FILE.FATAL=Erreur technique lors de la vérification globale du SIP)
    - Cas 3 : une erreur technique est survenue lors de la vérification globale du SIP (CHECK_SEDA.NOT_XSD_VALID.FATAL=Erreur technique lors de la vérification globale du SIP)

4.1.3.2. Vérification de l'en-tête du bordereau de transfert CHECK_HEADER (CheckHeaderActionHandler.java)

- **Règles** : tâche permettant de vérifier les informations générales du bordereau de transfert (nommées « header » dans le fichier « manifest.xml ») et de l'existence du service producteur (OriginatingAgencyIdentifier)
- **Type** : bloquant
- **Statuts** :
  - OK : les informations du bordereau de transfert sont conformes et le service producteur est déclaré (CHECK_HEADER.OK = Succès de la vérification générale du bordereau de transfert)

- KO :
  - Cas 1 : les informations du bordereau de transfert ne sont pas conformes ou il n'y a pas de service producteur déclaré (CHECK_HEADER.KO = Échec de la vérification générale du bordereau de transfert)
  - Cas 2 : les données référentielles sont inactives (CHECK_HEADER.INACTIVE.KO = Échec de la vérification générale du bordereau de transfert : donnée référentielle inactive)
  - Cas 3 : les données référentielles sont inconnues (CHECK_HEADER.UNKNOWN.KO = Échec de la vérification générale du bordereau de transfert : donnée référentielle inconnue)
  - Cas 4 : il y a une différence entre le profil déclaré dans le bordereau de transfert et celui déclaré dans le contrat (CHECK_HEADER.DIFF.KO = Échec de la vérification générale du bordereau de transfert : différence entre le profil déclaré dans le bordereau de transfert et celui déclaré dans le contrat)
  - Cas 5 : un des champs obligatoires n'est pas remplie (CHECK_HEADER.EMPTY_REQUIRED_FIELD.KO = Vérification générale du bordereau de transfert : champ obligatoire vide)
  - Cas 6 : la vérification a déjà été effectuée (CHECK_HEADER.ALREADY_EXECUTED = Action déjà exécutée : pas de vérification générale du bordereau de transfert)
- FATAL : une erreur technique est survenue lors des contrôles sur les informations générales du bordereau de transfert (CHECK_HEADER.FATAL = Erreur technique lors de la vérification générale du bordereau de transfert)

La tâche check_header contient les traitements suivants :

4.1.3.3. Vérification de la présence et contrôle des services agents CHECK_HEADER.CHECK_AGENT

Ce traitement n'est exécuté que si la valeur IN de *checkOriginatingAgency* est « true ».

- **Règle** : traitement consistant à vérifier le service producteur ainsi que le service versant déclarés dans le SIP par rapport au référentiel des services agents présent dans la solution logicielle Vitam
- **Type** : bloquant
- **Statuts** :
  - OK : le service producteur et, le cas échéant, le service versant déclarés dans le SIP est valide (service agent existant dans le référentiel des services agents)(CHECK_HEADER.CHECK_AGENT.OK = Succès de la vérification de la présence et du contrôle des services agents)
  - KO :
    - Cas 1 : aucun service producteur n'est déclaré dans la balise dédiée dans le bordereau de transfert (CHECK_HEADER.CHECK_AGENT.EMPTY_REQUIRED_FIELD.KO = Échec de la vérification de la présence et du contrôle des services agents : champ obligatoire vide)
    - Cas 2 : le service producteur et, le cas échéant, le service versant déclarés dans le SIP n'est pas connu du référentiel des services agents (CHECK_HEADER.CHECK_AGENT.UNKNOWN.KO = Échec de la vérification de la présence et du contrôle des services agents : services agents inconnus du référentiel des services agents)
    - Cas 3 : la balise permettant de déclarer un service producteur est absente du bordereau de tranfert (CHECK_HEADER.CHECK_AGENT.KO = Échec de la vérification de la présence et du contrôle des services agents)
  - FATAL : une erreur technique est survenue lors de la vérification de la présence et du contrôle des services agents (CHECK_HEADER.CHECK_AGENT.FATAL = Erreur technique lors de la vérification de la présence et du contrôle des services agents)

4.1.3.4. Vérification de la présence et contrôle du contrat d'entrée CHECK_HEADER.CHECK_CONTRACT_INGEST

Ce traitement n'est exécuté que si la valeur IN de *checkContract* est « true ».

- **Règle** : traitement consistant à vérifier le contrat d'entrée déclaré dans le SIP par rapport au référentiel des contrats d'entrée présent dans la solution logicielle Vitam
- **Type** : bloquant
- **Statuts** :
  - OK : le contrat déclaré dans le SIP existant dans le référentiel des contrats d'entrée de la solution logicielle Vitam et est actif (CHECK_HEADER.CHECK_CONTRACT_INGEST.OK = Succès de la vérification de la présence et du contrôle du contrat d'entrée)
  - KO :
    - Cas 1 : le contrat d'entrée déclaré dans le SIP est inexistant (CHECK_HEADER.CHECK_CONTRACT_INGEST.CONTRACT_UNKNOWN.KO = Échec de la vérification de la présence du contrat d'entrée : contrat d'entrée inconnu du référentiel des contrats d'entrée)
    - Cas 2 : le contrat d'entrée déclaré dans le SIP est inactif (CHECK_HEADER.CHECK_CONTRACT_INGEST.CONTRACT_INACTIVE.KO = Échec de la vérification du caractère actif du contrat d'entrée)
    - Cas 3 : aucun contrat d'entrée n'a été trouvé dans le manifeste (CHECK_HEADER.CHECK_CONTRACT_INGEST.CONTRACT_NOT_IN_MANIFEST.KO = Échec de la vérification de la présence du contrat d'entrée : le champ ArchivalAgreement est absent du bordereau de transfert)
    - Cas 4 : le contrat d'entrée déclaré dans le SIP n'existe pas dans le contexte applicatif (CHECK_HEADER.CHECK_CONTRACT_INGEST.CONTRACT_NOT_IN_CONTEXT.KO = Échec du contrôle de la présence du contrat d'entrée dans le contexte applicatif)
    - Cas 5 : le contexte applicatif est inexistant (CHECK_HEADER.CHECK_CONTRACT_INGEST.CONTEXT_UNKNOWN.KO = Échec du contrôle de la présence du contexte applicatif : contexte inconnu du référentiel des contextes)
    - Cas 6 : le contexte applicatif est inactif (CHECK_HEADER.CHECK_CONTRACT_INGEST.CONTEXT_INACTIVE.KO = Échec du contrôle du caractère actif du contexte applicatif)
    - Cas 7 : erreur lors de la récupération du contexte applicatif (CHECK_HEADER.CHECK_CONTRACT_INGEST.CONTEXT_CHECK_ERROR.KO = Échec de la vérification de la présence et du contrôle du contexte applicatif)
    - Cas 8 : le contrat de gestion déclaré est inexistant CHECK_HEADER.CHECK_CONTRACT_INGEST.MANAGEMENT_CONTRACT_UNKNOWN.KO = Échec de la vérification de la présence du contrat de gestion déclaré dans le contrat d'entrée: contrat de gestion connu dans le référentiel des contrats de gestion
    - Cas 9 : le contrat de gestion déclaré est inactif CHECK_HEADER.CHECK_CONTRACT_INGEST.MANAGEMENT_CONTRACT_INACTIVE.KO = Échec de la vérification de la présence du contrat de gestion déclaré dans le contrat d'entrée: contrat de gestion au statut inactif dans le référentiel des contrats de gestion
    - Cas 10 : le contrat de gestion déclaré est invalide CHECK_HEADER.CHECK_CONTRACT_INGEST.MANAGEMENT_CONTRACT_INVALID.KO = Échec de la vérification de la présence du contrat de gestion déclaré dans le contrat d'entrée: échec de validation des stratégies de stockage déclarées dans le contrat de gestion
  - FATAL : une erreur technique est survenue lors de la vérification de la présence et du contrôle du contrat d'entrée ou du contexte applicatif (CHECK_HEADER.CHECK_CONTRACT_INGEST.FATAL = Erreur technique lors de la vérification de la présence et du contrôle du contrat d'entrée ou du contexte applicatif)

4.1.3.5. Vérification de la relation entre le contrat d'entrée et le profil d'archivage CHECK_HEADER.CHECK_IC_AP_RELATION

Ce traitement n'est exécuté que si la valeur IN de *checkProfile* est « true ».

- **Règle** : traitement consistant à vérifier que le profil d'archivage déclaré dans le contrat d'entrée du SIP est le même que celui déclaré dans le bordereau de transfert.
- **Type** : bloquant
- **Statuts** :
  - OK : le profil d'archivage déclaré dans le contrat d'entrée et celui déclaré dans le bordereau de transfert sont les mêmes (CHECK_HEADER.CHECK_IC_AP_RELATION.OK = Succès de la vérification de la relation entre le contrat d'entrée et le profil)
  - KO :
    - Cas 1 : le profil d'archivage déclaré dans le SIP est inexistant (CHECK_HEADER.CHECK_IC_AP_RELATION.UNKNOWN.KO = Échec du contrôle de la présence du profil d'archivage dans le référentiel des profils d'archivage)
    - Cas 2 : le profil d'archivage déclaré dans le SIP est inactif (CHECK_HEADER.CHECK_IC_AP_RELATION.INACTIVE.KO = Échec du contrôle du caractère actif du profil d'archivage)
    - Cas 3 : le profil d'archivage déclaré dans le contrat d'entrée et celui déclaré dans le bordereau de transfert ne sont pas les mêmes (CHECK_HEADER.CHECK_IC_AP_RELATION.DIFF.KO = Échec du contrôle de cohérence entre le profil d'archivage déclaré dans le bordereau de transfert et celui déclaré dans le contrat d'entrée)
  - FATAL : une erreur technique est survenue lors de la vérification de la relation entre le contrat d'entrée et le profil d'archivage (CHECK_HEADER.CHECK_IC_AP_RELATION.FATAL = Erreur technique lors de la vérification de la relation entre le contrat d'entrée et le profil d'archivage)

4.1.3.6. Vérification de la conformité du bordereau de transfert par le profil d'archivage CHECK_HEADER.CHECK_ARCHIVEPROFILE

- **Règle** : traitement consistant à vérifier que le bordereau de transfert du SIP est conforme aux exigences du profil d'archivage. Si aucun profil d'archivage ne s'applique au SIP, ce traitement est ignoré
- **Type** : bloquant
- **Statuts** :
  - OK : le bordereau de transfert est conforme aux exigences du profil d'archivage (CHECK_HEADER.CHECK_ARCHIVEPROFILE.OK = Succès de la vérification de la conformité au profil d'archivage)
  - KO : le bordereau de transfert n'est pas conforme aux exigences du profil d'archivage (CHECK_HEADER.CHECK_ARCHIVEPROFILE.KO = Échec de la vérification de la conformité au profil d'archivage)
  - FATAL : une erreur technique est survenue lors de la vérification du bordereau de transfert par rapport au profil d'archivage (CHECK_HEADER.CHECK_ARCHIVEPROFILE.FATAL = Erreur technique lors de la vérification de la conformité au profil d'archivage)

4.1.3.7. Préparation des informations de stockage PREPARE_STORAGE_INFO (PrepareStorageInfoActionHandler.java)

- **Règle** : tâche consistant à récupérer les informations liées aux offres de stockage à partir des stratégies
- **Type** : bloquant
- **Statuts** :
  - OK : la récupération des informations de stockage a bien été effectuée (PREPARE_STORAGE_INFO.OK = Succès de la préparation des informations de stockage)
  - KO : la récupération des informations de stockage n'a pas pu être effectuée

(PREPARE_STORAGE_INFO.KO = Échec de la préparation des informations de stockage)

- FATAL : une erreur technique est survenue lors de la récupération des informations de stockage (PREPARE_STORAGE_INFO.FATAL = Erreur technique lors de la récupération des informations de stockage)

4.1.3.8. Vérification des objets et groupes d'objets CHECK_DATAOBJECTPACKAGE

- **Règle** : tâche consistant à vérifier les objets et groupes d'objets
- **Type** : bloquant
- **Statuts** :
  - OK : la vérification des objets et groupes d'objets à été effectué avec succès (CHECK_DATAOBJECTPACKAGE.OK=Succès de la vérification des objets et groupes d'objets)
  - KO : la vérification des objets et groupes d'objets n'a pu être effectué (CHECK_DATAOBJECTPACKAGE.KO=Échec de la vérification des objets et groupes d'objets)
  - FATAL : une erreur technique est servenue lors de la vérification des objets et groupes d'objets (CHECK_DATAOBJECTPACKAGE.FATAL=Erreur technique lors de la vérification des objets et groupes d'objets)

La tâche Check_DataObjectPackage contient les traitements suivants :

4.1.3.9. Vérification des usages des groupes d'objets CHECK_DATAOBJECTPACKAGE.CHECK_MANIFEST_DATAOBJECT_VERSION (CheckVersionActionHandler.java)

- **Règle** : traitement consistant à vérifier que tous les objets décrits dans le bordereau de transfert du SIP déclarent un usage conforme à la liste des usages acceptés dans la solution logicielle Vitam ainsi qu'un numéro de version respectant la norme de ce champ
- **Types d'usages acceptés :**
  - original papier (PhysicalMaster),
  - original numérique (BinaryMaster),
  - diffusion (Dissemination),
  - vignette (Thumbnail),
  - contenu brut (TextContent).

  Les numéros de versions sont optionnels, il s'agit d'un entier positif ou nul (0, 1, 2…).

  La grammaire est : « usage_version ». Exemples : « BinaryMaster_2 », « TextContent_10 » ou sans numéro de versions « PhysicalMaster ».
- **Type** : bloquant
- **Statuts** :
  - OK : les objets contenus dans le SIP déclarent tous dans le bordereau de transfert un usage cohérent avec ceux acceptés et optionnellement un numéro de version respectant la norme de ce champ usage, par exemple « BinaryMaster_2 » (CHECK_DATAOBJECTPACKAGE.CHECK_MANIFEST.DATAOBJECT.VERSION.OK = Succès de la vérification des usages des objets)
  - KO :
    - Cas 1 : un ou plusieurs BinaryMaster sont déclarés dans un ou plusieurs objets physiques (CHECK_DATAOBJECTPACKAGE.CHECK_MANIFEST_DATAOBJECT_VERSION.PDO_DATAOBJECTIONVERSION_BINARYMASTER.KO = L'objet physique déclare un usage « BinaryMaster ». Cet usage n'est pas autorisé pour les objets physiques
    - Cas 2 : un ou plusieurs PhysicalMaster sont déclarés dans un ou plusieurs objets binaires (CHECK_DATAOBJECTPACKAGE.BDO_DATAOBJECTIONVERSION_PHYSICALMASTER.KO = Au moins un objet binaire déclare un usage « PhysicalMaster ». Cet usage n'est pas autorisé pour les objets binaires)

- Cas 3 : un ou plusieurs objets contenus dans le SIP déclarent dans le bordereau de transfert un usage ou un numéro de version incohérent avec ceux acceptés (CHECK_DATAOBJECTPACKAGE.CHECK_MANIFEST_DATAOBJECT_VERSION.INVALID_DATAOBJECTVERSION.KO = Cet objet déclare un usage incorrect. L'usage doit s'écrire sous la forme [usage] ou [usage]_[version]. « Usage » doit être parmi l'énumération DataObjectVersion définie pour Vitam, « version » doit être un entier positif)
- Cas 4 : une ou plusieurs URI sont vides (CHECK_DATAOBJECTPACKAGE.CHECK_MANIFEST_DATAOBJECT_VERSION.EMPTY_REQUIRED_FIELD.KO = Il existe au moins un champ non renseigné dont la valeur est obligatoire)

o FATAL : une erreur technique est survenue lors du contrôle des usages déclarés dans le bordereau de transfert pour les objets contenus dans le SIP (CHECK_MANIFEST_DATAOBJECT_VERSION.FATAL = Erreur technique lors de la vérification des usages des objets)

4.1.3.10. Vérification du nombre d'objets CHECK_DATAOBJECTPACKAGE.CHECK_MANIFEST_OBJECTNUMBER (CheckObjectsNumberActionHandler.java)

- **Règle** : traitement consistant à vérifier que le nombre d'objets binaires reçus dans la solution logicielle Vitam et stocké dans l'espace de travail interne (« workspace ») est strictement égal au nombre d'objets binaires déclaré dans le manifeste du SIP
- **Type** : bloquant
- **Statuts** :
  - OK : le nombre d'objets reçus dans la solution logicielle Vitam est strictement égal au nombre d'objets déclarés dans le bordereau de transfert du SIP (CHECK_DATAOBJECTPACKAGE.CHECK_MANIFEST_OBJECTNUMBER.OK = Succès de la vérification du nombre d'objets)
  - KO :
    - Cas 1 : le nombre d'objets reçus dans la solution logicielle Vitam est supérieur au nombre d'objets déclaré dans le bordereau de transfert du SIP (CHECK_DATAOBJECTPACKAGE.CHECK_MANIFEST_OBJECTNUMBER.MANIFEST_INFERIOR_BDO.KO = Le bordereau de transfert déclare moins d'objets binaires qu'il n'en existe dans le répertoire Content du SIP)
    - Cas 2 : le nombre d'objets reçus dans la solution logicielle Vitam est inférieur au nombre d'objets déclaré dans le bordereau de transfert du SIP (CHECK_DATAOBJECTPACKAGE.CHECK_MANIFEST_OBJECTNUMBER.MANIFEST_SUPERIOR_BDO.KO = Le bordereau de transfert déclare plus d'objets binaires qu'il n'en existe dans le répertoire Content du SIP)
    - Cas 3 : une ou plusieurs balises URI déclarent un chemin invalide (CHECK_DATAOBJECTPACKAGE.CHECK_MANIFEST_OBJECTNUMBER.INVALID_URI.KO = Au moins un objet déclare une URI à laquelle ne correspond pas de fichier ou déclare une URI déjà utilisée par un autre objet)
  - FATAL : une erreur technique est survenue lors de la vérification du nombre d'objets (CHECK_DATAOBJECTPACKAGE.CHECK_MANIFEST_OBJECTNUMBER.FATAL = Erreur technique lors de la vérification du nombre d'objets)

4.1.3.11. Vérification de la cohérence du bordereau de transfert CHECK_DATAOBJECTPACKAGE.CHECK_MANIFEST (ExtractSedaActionHandler.java)

- **Règle** : traitement consistant à :
  - créer les journaux du cycle de vie des unités archivistiques et des groupes d'objets,

  - extraire les unités archivistiques, objets binaires et objets physiques du bordereau de transfert,
  - vérifier la présence de récursivité dans les arborescences des unités archivistiques et à créer l'arbre d'ordre d'indexation,
  - extraire les métadonnées contenues dans la balise ManagementMetadata du bordereau de transfert pour le calcul des règles de gestion,
  - vérifier la validité des rattachements des unités du SIP aux unités présentes dans la solution logicielle Vitam si demandés,
  - détecter des problèmes d'encodage dans le bordereau de transfert et vérifier que les objets ne font pas référence directement à des unités si ces objets possèdent des groupes d'objets,
  - vérifier la présence obligatoire d'un objet de type Master pour une entrée, et vérifier les usages d'objets autorisés pour les rattachements.
- **Type** : bloquant
- **Statuts** :
  - OK : les journaux du cycle de vie des unités archivistiques et des groupes d'objets ont été créés avec succès, aucune récursivité n'a été détectée dans l'arborescence des unités archivistiques, la structure de rattachement déclarée existe, le type de structure de rattachement est autorisé, (par exemple, un SIP peut être rattaché à un plan de classement, mais pas l'inverse) aucun problème d'encodage n'a été détecté et les objets avec groupe d'objets ne référencent pas directement les unités. L'extraction des unités archivistiques, objets binaires et physiques, la création de l'arbre d'indexation et l'extraction des métadonnées des règles de gestion ont été effectuées avec succès, les vérifications au niveau des types d'usages autorisés ont bien été effectués.
    (CHECK_DATAOBJECTPACKAGE.CHECK_MANIFEST.OK = Succès du contrôle de cohérence du bordereau de transfert).
  - KO :
    - Cas 1 : au moins une demande de rattachement à des unités archivistiques existantes dans le système a échoué, car le nœud de rattachement déclaré dans le contrat d'entrées a pour valeur « null »
      (CHECK_DATAOBJECTPACKAGE.CHECK_MANIFEST.NULL_LINK_PARENT_ID_ATTACHMENT.KO = Le rattachement n'a pas été effectué : le contrat d'entrée ne déclare pas de nœud de rattachement)
    - Cas 2 : au moins une demande de rattachement à des unités archivistiques existantes dans le système a échoué, car le contrat d'entrée requiert un rattachement à au moins une unité archivistique
      (CHECK_DATAOBJECTPACKAGE.CHECK_MANIFEST.ATTACHMENT_REQUIRED.KO = Le contrat d'entrée requiert un rattachement à au moins une unité archivistique)
    - Cas 3 : au moins une demande de rattachement à des unités archivistiques existantes dans le système a échoué, car le contrat d'entrée n'autorise pas les rattachements
      (CHECK_DATAOBJECTPACKAGE.CHECK_MANIFEST.UNAUTHORIZED_ATTACHMENT_BY_CONTRACT.KO = Le rattachement n'a pas été effectué : le contrat d'entrée n'autorise pas les rattachements)
    - Cas 4 : au moins une demande de rattachement à des unités archivistiques existantes dans le système a échoué en raison d'un nombre d'unités archivistiques répondant à la requête effectuée supérieur à 1
      (CHECK_DATAOBJECTPACKAGE.CHECK_MANIFEST.TOO_MANY_FOUND_ATTACHMENT.KO = Le rattachement n'a pas été effectué : l'élément de rattachement n'est pas unique dans le système)
    - Cas 5 : au moins un objet binaire dans le bordereau de transfert déclare plusieurs version d'un même usage
      (CHECK_DATAOBJECTPACKAGE.CHECK_MANIFEST.TOO_MANY_VERSION_BY_USAGE.KO = Le transfert de plusieurs versions d'un même usage dans un même versement est interdit)
    - Cas 6 : au moins une demande de rattachement à des unités archivistiques existantes dans le

système a échoué en raison d'un nombre d'unités archivistiques répondant à la requête, égal à 0 (CHECK_DATAOBJECTPACKAGE.CHECK_MANIFEST.NOT_FOUND_ATTACHMENT.KO = Le rattachement n'a pas été effectué : l'élément de rattachement n'existe pas dans le système)

- Cas 7 : au moins une demande de rattachement à des unités archivistiques existantes dans le système a échoué, car le rattachement demandé n'est pas autorisé (CHECK_DATAOBJECTPACKAGE.CHECK_MANIFEST.UNAUTHORIZED_ATTACHMENT.KO = Le rattachement n'a pas été effectué : le rattachement n'est pas situé dans le périmètre autorisé)
- Cas 8 : au moins une demande de rattachement à des unités archivistiques existantes dans le système a échoué, car le GUID déclaré n'est pas valide (CHECK_DATAOBJECTPACKAGE.CHECK_MANIFEST.INVALID_GUID_ATTACHMENT.KO = Le rattachement n'a pas été effectué : l'élément de rattachement est incorrect)
- Cas 9 : au moins une demande de rattachement à des unités archivistiques existantes dans le système a échoué, car elle provoquerait une récursivité de l'arborescence (CHECK_DATAOBJECTPACKAGE.CHECK_MANIFEST.MODIFY_PARENT_EXISTING_UNIT_UNAUTHORIZED.KO = Le rattachement n'a pas été effectué : impossibilité de rattacher une unité archivistique existante à une unité archivistique parente)
- Cas 10 : une ou plusieurs balises de rattachement vers un groupe d'objets techniques existant déclarent autre chose que le GUID d'un groupe d'objets techniques existant (CHECK_DATAOBJECTPACKAGE.CHECK_MANIFEST.EXISTING_OG_NOT_DECLARED.KO = Une unité archivistique déclare un objet à la place du groupe d'objets correspondant)
- Cas 11 : une récursivité a été détectée dans l'arborescence des unités archivistiques (CHECK_DATAOBJECTPACKAGE.CHECK_MANIFEST.CHECK_MANIFEST_LOOP.KO = Le bordereau de transfert présente une récursivité dans l'arborescence de ses unités archivistiques)
- Cas 12 : il y a un problème d'encodage ou des objets référencent directement des unités archivistiques (CHECK_DATAOBJECTPACKAGE.CHECK_MANIFEST.KO = Échec du contrôle de cohérence du bordereau de transfert)
- Cas 13 : présence attendue d'un objet de type Master: Binary ou physical CHECK_DATAOBJECTPACKAGE.CHECK_MANIFEST.MASTER_MANDATORY_REQUIRED.KO = Absence d'un BinaryMaster ou PhysicalMaster dans le groupe d'objet
- Cas 14 : le contrat d'entrée n'autorise pas un ou plusieurs usages d'objets (CHECK_DATAOBJECTPACKAGE.CHECK_MANIFEST.ATTACHMENT_OBJECTGROUP.KO = Le contrat d'entrée n'autorise pas le rattachement d'un objet à un groupe d'objets existant)
- Cas 15 : il y a une donnée malformatée (CHECK_DATAOBJECTPACKAGE.CHECK_MANIFEST_MALFORMED_DATA.KO = Le bordereau de transfert possède une donnée malformée)

o FATAL : une erreur technique est survenue lors de la vérification de la cohérence du bordereau, par exemple les journaux du cycle de vie n'ont pu être créés (CHECK_MANIFEST.FATAL = Erreur technique lors du contrôle de cohérence du bordereau de transfert)

4.1.3.12. Vérification de la cohérence entre objets, groupes d'objets et unités archivistiques CHECK_DATAOBJECTPACKAGE.CHECK_CONSISTENCY (CheckObjectUnitConsistencyActionHandler.java)

- **Règle** : traitement consistant à vérifier que chaque objet ou groupe d'objets est référencé par une unité archivistique, à rattacher à un groupe d'objets les objets sans groupe d'objets mais référencés par une unité archivistique, à créer la table de concordance (MAP) entre les identifiants des objets et des unités archivistiques du SIP et à générer leurs identifiants pérennes dans la solution logicielle Vitam (GUID)
- **Type** : bloquant
- **Statuts** :
  - o OK : aucun objet ou groupe d'objets n'est orphelin (c'est-à-dire non référencé par une unité archivistique) et tous les objets sont rattachés à un groupe d'objets. La table de concordance est créée

et les identifiants des objets et unités archivistiques ont été générés (CHECK_DATAOBJECTPACKAGE.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'objets est orphelin (c'est-à-dire non référencé par une unité archivistique) (CHECK_DATAOBJECTPACKAGE.CHECK_CONSISTENCY.KO = Échec de la vérification de la cohérence entre objets, groupes d'objets et unités archivistiques)
- FATAL : une erreur technique est survenue lors de la vérification de la cohérence entre objets, groupes d'objets et unités archivistiques (CHECK_DATAOBJECTPACKAGE.CHECK_CONSISTENCY.FATAL = Erreur technique lors de la vérification de la cohérence entre objets, groupes d'objets et unités archivistiques)

4.1.3.13. Vérification du rattachement à un groupe d'objets ou une unité archivistique entrés sans erreur CHECK_ATTACHEMENT (CheckAttachementActionHandler.java)

- **Règle** : tâche consistant à vérifier le rattachement à des groupes d'objets techniques et des unités archivistiques entrés sans erreur dans le système.
- **Type** : bloquant
- **Statuts** :
  - OK : la vérification des objets et groupes d'objets à été effectué avec succès (CHECK_ATTACHEMENT.OK=Succès de la vérification du rattachement entre objets, groupes d'objets et unités archivistiques existantes et les nouveaux)
  - KO : la vérification des objets et groupes d'objets n'a pu être effectuée, car le groupe d'objets techniques ou l'unité archivistique devant faire l'objet d'un rattachement dans le système sont entrés en erreur (CHECK_ATTACHEMENT.KO=Échec de la vérification du rattachement entre objets, groupes d'objets et unités archivistiques existantes et les nouveaux)
  - FATAL : une erreur technique est survenue lors de la vérification des objets et groupes d'objets (CHECK_ATTACHEMENT.FATAL=Erreur technique lors de la vérification du rattachement entre objets, groupes d'objets et unités archivistiques existantes et les nouveaux)

4.1.4.Processus de contrôle et traitement des objets (STP_OG_CHECK_AND_TRANSFORME)

4.1.4.1. Vérification de l'intégrité des objets CHECK_DIGEST (CheckConformityActionPlugin.java)

- **Règle** : tâche consistant à vérifier la cohérence entre l'empreinte de l'objet binaire calculée par la solution logicielle Vitam et celle déclarée dans le bordereau de transfert. Si l'empreinte déclarée dans le bordereau de transfert n'a pas été calculée avec l'algorithme SHA-512, alors l'empreinte est recalculée avec cet algorithme. Elle sera alors enregistrée dans la solution logicielle Vitam.

  Algorithmes autorisés en entrée : MD5, SHA-1, SHA-256, SHA-512
- **Type** : bloquant
- **Statuts** :
  - OK : tous les objets binaires reçus sont identiques aux objets binaires attendus. Tous les objets binaires disposent désormais d'une empreinte calculée avec l'algorithme SHA-512 (CHECK_DIGEST.OK = Succès de la vérification de l'empreinte des objets)
  - KO :
    - Cas 1 : au moins un objet reçu n'a pas d'empreinte dans le bordereau (CHECK_DIGEST.EMPTY.KO = Échec lors de la vérification de l'empreinte des objets : Il existe au moins un objet dont l'empreinte est absente dans le bordereau de transfert)
    - Cas 2 : au moins une empreinte d'un objet reçu n'est pas conforme à son empreinte dans le

bordereau (CHECK_DIGEST.INVALID.KO = Échec lors de la vérification de l'empreinte des objets : Il existe au moins un objet dont l'empreinte est invalide dans le bordereau de transfert)

- Cas 3 : le SIP soumis à la solution logicielle Vitam contient à la fois le cas 1 et le cas 2 (CHECK_DIGEST.KO = Échec de la vérification de l'empreinte des objets)

- FATAL : une erreur technique est survenue lors de la vérification de l'intégrité des objets binaires, par exemple lorsque l'algorithme est inconnu (CHECK_DIGEST.FATAL = Erreur technique lors de la vérification de l'empreinte des objets)

4.1.4.2. Calcul de la taille des fichiers CHECK_OBJECT_SIZE (CheckObjectSizeActionPlugin.java)

- **Règle** : tâche vérifier la taille de chaque objet binaire présent dans le SIP, à vérifier que la taille des objets correspond à la taille des fichiers renseignée pour chacun d'eux dans le manifeste. Le poids des fichiers est calculé en octets et comparé à la taille renseignée dans le manifeste. En cas d'incohérence entre la déclaration dans le manifeste et la taille du fichier, le SIP sera accepté, générant un avertissement. La solution logicielle Vitam se servira alors des informations qu'elle a identifiées et non de celles fournies dans le SIP
- **Type** : bloquant
- **Statuts** :
  - OK (CHECK_OBJECT_SIZE.OK = Succès de la vérification de la taille des objets) :
    - la taille des fichiers correspond à celle qui est renseignée dans le manifeste et aucune incohérence n'a été trouvée ;
    - la taille des fichiers n'est pas renseignée dans le manifeste et la solution logicielle Vitam enregistre la taille des fichiers qu'elle a calculée.
  - WARNING (CHECK_OBJECT_SIZE.WARNING = Avertissement de la vérification de la taille des objets) : au moins un objet reçu a une taille renseignée dans le manifeste qui n'est pas identique à celle des fichiers numériques.
  - FATAL : une erreur technique est survenue lors de la vérification de la taille des objets (CHECK_OBJECT_SIZE .FATAL = Erreur technique lors de la vérification de la taille des objets)

4.1.4.3. Identification des formats OG_OBJECTS_FORMAT_CHECK (FormatIdentificationActionPlugin.java)

- **Règle** : tâche consistant à identifier le format de chaque objet binaire présent dans le SIP, à vérifier que le format identifié des objets correspond à la liste des formats acceptés dans le contrat d'entrée et à vérifier que le format identifié des objets est référencé dans le référentiel des formats de la solution logicielle Vitam. 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 avec les formats enregistrés dans le référentiel des formats interne à la solution logicielle Vitam et avec celles déclarées dans le bordereau de transfert. En cas d'incohérence entre la déclaration dans le SIP et le format identifié, le SIP sera accepté, générant un avertissement. La solution logicielle Vitam se servira alors des informations qu'elle a identifiées et non de celles fournies dans le SIP
- **Type** : bloquant
- **Statuts** :
  - OK : l'identification s'est bien passée, les formats ont tous été identifiés, sont référencés dans le référentiel interne et sont soit dans la liste des formats autorisés du contrat d'entrée, soit ce contrat autorise tous les formats. De plus les informations de formats trouvées par la solution logicielle Vitam sont cohérentes avec celles déclarées dans le manifeste (OG_OBJECTS_FORMAT_CHECK.OK =

Succès de la vérification des formats)

- KO :
  - Cas 1 : au moins un objet reçu a un format qui n'a pas été trouvé et le contrat d'entrée utilisé interdit le versement d'objets aux formats non identifiés (OG_OBJECTS_FORMAT_CHECK.KO = Échec de l'identification des formats)
  - Cas 2 : au moins un objet reçu a un format qui n'est pas référencé dans le référentiel interne (OG_OBJECTS_FORMAT_CHECK.UNCHARTED.KO = Échec de l'identification des formats, le format de ou des objet(s) est identifié mais est inconnu du référentiel des formats)
  - Cas 3 : au moins objet reçu possède un format qui n'est pas indiqué dans la liste des formats autorisés du contrat d'entrée du SIP (OG_OBJECTS_FORMAT_CHECK.REJECTED_FORMAT.KO = Échec de l'identification des formats : le contrat d'entrée interdit le versement d'objet au format inconnu et le SIP versé contient au moins un objet au format inconnu, ou bien le SIP contient un format interdit par le contrat d'entrée)
- WARNING :
  - Cas 1 : 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 l'identification des formats)
  - Cas 2 : au moins un objet reçu a un format qui n'a pas été trouvé mais le contrat d'entrée utilisé autorise le versement d'objets aux formats non identifiés. Dans ce cas Vitam remplace le champ « FormatId » du manifest.xml par le mot « unknown » (OG_OBJECTS_FORMAT_CHECK.WARNING = Avertissement lors de l'identification des formats)
- FATAL : une erreur technique est survenue lors de l'identification des formats (OG_OBJECTS_FORMAT_CHECK.FATAL = Erreur technique lors de l'identification des formats)

4.1.5.Processus de contrôle et traitement des unités archivistiques (STP_UNIT_CHECK_AND_PROCESS)

4.1.5.1. Vérification globale de l'unité archivistique CHECK_UNIT_SCHEMA (CheckArchiveUnitSchemaActionPlugin.java)

- **Règle** : tâche consistant à :
  - contrôler que la valeur des champs déclarés dans le bordereau de transfert est d'un type conforme à celui déclaré dans l'ontologie ;
  - contrôler la validité des champs de l'unité archivistique par rapport au schéma prédéfini dans la solution logicielle Vitam. Par exemple, les champs obligatoires, comme les titres des unités archivistiques, ne doivent pas être vides. Lorsque le manifeste déclare une personne (Person) et non une société (Entity), alors au moins un champ entre « Firstname » et « Birthname » est obligatoire,
  - vérifier que la date de fin est bien supérieure ou égale à la date de début de l'unité archivistique.
- **Type** : bloquant
- **Statuts** :
  - OK : tous les champs de l'unité archivistique sont conformes à ce qui est attendu (CHECK_UNIT_SCHEMA.OK = Succès de la vérification globale de l'unité archivistique)
  - KO :
    - Cas 1 : au moins un champ d'une unité archivistique déclare un champ dont la valeur n'est pas conforme au type défini dans l'ontologie (CHECK_UNIT_SCHEMA.KO = Échec de la vérification globale de l'unité archivistique)

- Cas 2 : au moins un champ d'une unité archivistique dont le schéma n'est pas conforme par rapport au schéma par défaut des unités archivistiques défini pour la solution logicielle Vitam. (CHECK_UNIT_SCHEMA.INVALID_UNIT.KO = Échec lors de la vérification globale de l'unité archivistique : champs non conformes)
- Cas 3 : au moins un champ obligatoire d'une unité archivistique est vide (CHECK_UNIT_SCHEMA.EMPTY_REQUIRED_FIELD.KO = Échec lors de la vérification globale de l'unité archivistique : champs obligatoires vides)
- Cas 4 : au moins un champ date d'une unité archivistique est supérieur à 9000 ou la date de fin des dates extrêmes est strictement inférieure à la date de début (CHECK_UNIT_SCHEMA.RULE_DATE_THRESHOLD.KO = Échec du calcul des dates d'échéance, la date ne peut être gérée)
- Cas 5 : au moins un champ date d'une unité archivistique déclare une valeur non conforme au type attendu (CHECK_UNIT_SCHEMA.RULE_DATE_FORMAT.KO = Échec du calcul des dates d'échéance, la date ne peut être gérée)
- Cas 6 : au moins une valeur de l'unité archivistique n'est pas conforme à son schéma en raison d'un problème de cohérence entre champs. Par exemple, la valeur contenue dans le champ « StartDate » est postérieure à la date définie dans la « EndDate » (CHECK_UNIT_SCHEMA.CONSISTENCY.KO = Au moins une unité archivistique n'est pas conforme à son schéma en raison d'un problème de cohérence entre champs)

o FATAL : une erreur technique est survenue lors de la vérification de l'unité archivistique (CHECK_UNIT_SCHEMA.FATAL = Erreur technique lors de la vérification globale de l'unité archivistique)

4.1.5.2. Vérification du profil d'unité archivistique CHECK_ARCHIVE_UNIT_PROFILE (CheckArchiveUnitProfileActionPlugin.java)

- **Règle** : tâche consistant à vérifier la conformité des unités archivistiques au schéma défini dans les profils d'unités archivistiques qu'elles déclarent dans la balise « ArchiveUnitProfile ». Les profils d'unités archivistiques référencés doivent être en état « Actif » et ne pas avoir un schéma de contrôle vide
- **Type** : non bloquant
- **Statuts** :
  - OK : les unités archivistiques déclarant un profil d'unité archivistique de référence sont bien conformes au schéma décrit dans le profil d'unité archivistique, et le profil et le schéma existent bien dans le système en état actif (CHECK_ARCHIVE_UNIT_PROFILE.OK = Succès de la vérification de la conformité au profil d'unité archivistique)
  - KO :
    - Cas 1 : au moins une unité archivistique n'est pas conforme au schéma décrit dans le profil d'unité archivistique associé (CHECK_ARCHIVE_UNIT_PROFILE.KO = Échec de la vérification de la conformité au profil d'unité archivistique)
    - Cas 2 : au moins une unité archivistique qui déclare un lien avec un profil d'unité archivistique inexistant dans le référentiel (CHECK_ARCHIVE_UNIT_PROFILE.PROFILE_NOT_FOUND.KO = Échec de la vérification de la conformité au profil d'unité archivistique : profil d'unité archivistique non trouvé)
    - Cas 3 : au moins une unité archivistique qui n'est pas conforme au schéma décrit dans le profil d'unité archivistique associé (CHECK_ARCHIVE_UNIT_PROFILE.INVALID_UNIT.KO = Échec de la vérification de la conformité au profil d'unité archivistique : champs non conformes)
    - Cas 4 : le profil d'unité archivistique cité dans le référentiel est mal formaté (CHECK_ARCHIVE_UNIT_PROFILE.INVALID_AU_PROFILE.KO = Échec de la vérification de la conformité aux documents type : profil d'unité archivistique non conforme)
    - Cas 5 : le profil d'unité archivistique est dans l'état « inactif » (CHECK_ARCHIVE_UNIT_PROFILE.INACTIVE_STATUS.KO = Échec de la vérification de la conformité aux documents type : profil d'unité archivistique au statut « inactif »

- Cas 6 : le profil d'unité archivistique possède un schéma de contrôle qui est vide (CHECK_ARCHIVE_UNIT_PROFILE.EMPTY_CONTROL_SCHEMA.KO = Échec de la vérification de la conformité aux documents type : schéma de contrôle du profil d'unité archivistique vide)

4.1.5.3. Vérification du niveau de classification CHECK_CLASSIFICATION_LEVEL (CheckClassificationLevelActionPlugin.java)

- **Règle** : tâche consistant à vérifier les niveaux de classification associés, s'il en existe, aux unités archivistiques. Ces niveaux doivent exister dans la liste des niveaux de classification autorisés par la plateforme (paramètre configuré dans la configuration des workers). Pour les unités archivistiques sans niveau de classification, la vérification contrôle que la plateforme autorise le versement d'unités archivistiques ne déclarant pas de niveau de classification.
- **Type** : bloquant
- **Statuts** :
  - OK : les unités archivistiques versées ont un niveau de classification autorisé par la plateforme. S'il existe dans le SIP des unités archivistiques sans niveau de classification, il faut que la plateforme autorise le versement d'unités archivistiques sans niveau de classification. (CHECK_CLASSIFICATION_LEVEL.OK = Succès de la vérification du niveau de classification)
  - KO : au moins une unité archivistique du SIP possède un niveau de classification qui n'est pas un niveau de classification autorisé par la plateforme, ou une unité archivistique n'a pas de niveau de classification alors que la plateforme requiert que toutes les unités archivistiques possèdent un niveau de classification. (CHECK_CLASSIFICATION_LEVEL.KO = Échec de la vérification du niveau de classification, non autorisé par la plateforme : le bordereau de transfert déclare un niveau de classification non autorisé par la plateforme)
  - FATAL : une erreur technique est survenue lors de la vérification des niveaux de classification (CHECK_CLASSIFICATION_LEVEL.FATAL = Erreur technique lors de la vérification du niveau de classification)

4.1.5.4. Application des règles de gestion et calcul des dates d'échéances UNITS_RULES_COMPUTE (UnitsRulesComputePlugin.java)

- **Règle** : tâche consistant à calculer les dates d'échéances des unités archivistiques du SIP. Pour les unités racines, c'est-à-dire les unités déclarées dans le SIP et n'ayant aucun parent dans l'arborescence, la solution logicielle Vitam utilise les règles de gestion incluses dans le bloc Management de chacune de ces unités ainsi que celles présentes dans le bloc ManagementMetadata. La solution logicielle Vitam effectue également ce calcul pour les autres unités archivistiques du SIP possédant des règles de gestion déclarées dans leurs balises Management, sans prendre en compte le ManagementMetadata. Le référentiel utilisé pour ces calculs est le référentiel des règles de gestion de la solution logicielle Vitam.
- **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 de l'application des règles de gestion et du calcul des dates d'échéance)
  - KO :
    - Cas 1 : au moins une unité archivistique déclare un champ dont la valeur nest pas conforme à celle attendue (UNITS_RULES_COMPUTE.KO=Au moins une unité archivistique déclare un champ dont la valeur n'"est pas conforme à celle attendue)
    - Cas 2 : au moins une unité archivistique déclare une règle non référencée dans le référentiel interne (UNITS_RULES_COMPUTE.UNKNOWN.KO = Échec lors de l'application des règles de gestion et du calcul des dates d'échéance : règle de gestion inconnue)
    - Cas 3 : au moins une unité archivistique déclare une règle non cohérente avec sa catégorie (UNITS_RULES_COMPUTE.CONSISTENCY.KO = Échec lors de l'application des règles de

gestion et du calcul des dates d'échéance : Au moins une unité archivistique déclare une règle non cohérente avec sa catégorie)

- Cas 4 : au moins une unité archivistique déclare dans le champ RefNonRuleId une règle non cohérente avec sa catégorie (UNITS_RULES_COMPUTE.REF_INCONSISTENCY.KO = Échec lors de l'application des règles de gestion et du calcul des dates d'échéance : exclusion d'héritage incohérente)

- FATAL : une erreur technique est survenue lors du calcul des dates d'échéances (UNITS_RULES_COMPUTE.FATAL = Erreur technique lors de l'application des règles de gestion et du calcul des dates d'échéance)

4.1.6.Processus de vérification préalable à la prise en charge (STP_STORAGE_AVAILABILITY_CHECK)

4.1.6.1. Vérification de la disponibilité de toutes les offres de stockage STORAGE_AVAILABILITY_CHECK (CheckStorageAvailabilityActionHandler.java)

- **Règle** : tâche consistant à vérifier la disponibilité des offres de stockage et de l'espace disponible pour y stocker le contenu du SIP compte tenu de la taille des objets à stocker
- **Type** : bloquant
- **Statuts** :
  - OK : les offres de stockage sont accessibles et disposent d'assez d'espace pour stocker le contenu du SIP (STORAGE_AVAILABILITY_CHECK.OK = Succès de la vérification de la disponibilité de toutes les offres de stockage)
  - KO :
    - Cas 1 : les offres de stockage ne sont pas disponibles (STORAGE_AVAILABILITY_CHECK.STORAGE_OFFER_KO_UNAVAILABLE.KO = Échec de la vérification de la disponibilité d'au moins une offre de stockage)
    - Cas 2 : les offres ne disposent pas d'assez d'espace pour stocker le contenu du SIP (STORAGE_AVAILABILITY_CHECK.STORAGE_OFFER_SPACE_KO.KO = Échec de la vérification de l'espace disponible)
  - FATAL : une erreur technique est survenue lors de la vérification de la disponibilité de l'offre de stockage (STORAGE_AVAILABILITY_CHECK.FATAL = Erreur technique lors de la vérification de la disponibilité d'au moins une offre de stockage)

La tâche Check_Availability_Check contient le traitement suivant :

4.1.6.2. Vérification de la disponibilité de l'offre de stockage STORAGE_AVAILABILITY_CHECK.STORAGE_AVAILABILITY_CHECK (CheckStorageAvailabilityActionHandler.java)

- **Règle** : traitement consistant à vérifier la disponibilité de l'offre de stockage et de l'espace disponible pour y stocker le contenu du SIP compte tenu de la taille des objets à stocker
- **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.STORAGE_AVAILABILITY_CHECK.OK = Succès de la vérification de la disponibilité de l'offre de stockage)
  - KO :
    - Cas 1 : l'offre de stockage n'est pas disponible (STORAGE_AVAILABILITY_CHECK.STORAGE_AVAILABILITY_CHECK.STORAGE_OFFER_KO_UNAVAILABLE.KO = L'offre de stockage n'est pas disponible)

- Cas 2 : l'offre de stockage ne dispose pas d'assez d'espace pour stocker le contenu du SIP (STORAGE_AVAILABILITY_CHECK.STORAGE_AVAILABILITY_CHECK.STORAGE_OFFER_SPACE_KO.KO = Disponibilité de l'offre de stockage insuffisante)

- FATAL : une erreur technique est survenue lors de la vérification de la disponibilité de l'offre de stockage (STORAGE_AVAILABILITY_CHECK.STORAGE_AVAILABILITY_CHECK.FATAL = Erreur technique lors de la vérification de la disponibilités de l'offre de stockage)

4.1.7. Processus d'écriture et indexation des objets et groupes d'objets (STP_OBJ_STORING)

4.1.7.1. Écriture des objets sur l'offre de stockage OBJ_STORAGE (StoreObjectActionHandler.java)

- **Règle** : tâche consistant à écrire les objets contenus dans le SIP sur les offres de stockage en fonction de la stratégie de stockage applicable
- **Type** : Bloquant
- **Statuts** :
  - OK : tous les objets binaires contenus dans le SIP ont été écrits sur les offres de stockage (OBJ_STORAGE.OK = Succès de l'écriture des objets et des groupes d'objets sur les offres de stockage)
  - KO : au moins un des objets binaires contenus dans le SIP n'a pas pu être écrit sur les offres de stockage (OBJ_STORAGE.KO = Échec de l'écriture des objets et des groupes d'objets sur les offres de stockage)
  - WARNING : le SIP ne contient pas d'objet (OBJECTS_LIST_EMPTY.WARNING = Avertissement lors de l'établissement de la liste des objets : il n'y a pas d'objet pour cette étape)
  - FATAL : une erreur technique est survenue lors de l'écriture des objets binaires sur les offres de stockage (OBJ_STORAGE.FATAL = Erreur technique lors de l'écriture des objets et des groupes d'objets sur les offres de stockage)

4.1.7.2. Indexation des métadonnées des groupes d'objets et objets OG_METADATA_INDEXATION (IndexObjectGroupActionPlugin.java)

- **Règle** : tâche consistant à indexer les métadonnées des groupes d'objets et objets dans les bases internes de la solution logicielle Vitam, comme la taille des objets, les métadonnées liées aux formats (Type MIME, PUID, etc.), l'empreinte des objets, etc.
- **Type** : bloquant
- **Statuts** :
  - OK : les métadonnées des groupes d'objets et objets ont été indexées avec succès (OG_METADATA_INDEXATION.OK = Succès de l'indexation des métadonnées des objets et des groupes d'objets)
  - KO : au moins une des métadonnées des groupes d'objets et objets n'a pas été indexée (OG_METADATA_INDEXATION.KO = Échec de l'indexation des métadonnées des objets et des groupes d'objets)
  - FATAL : une erreur technique est survenue lors de l'indexation des métadonnées des groupes d'objets (OG_METADATA_INDEXATION.FATAL = Erreur technique lors de l'indexation des métadonnées des objets et des groupes d'objets)

4.1.8.Processus d'indexation des unités archivistiques (STP_UNIT_METADATA)

4.1.8.1. Indexation des métadonnées des unités archivistiques UNIT_METADATA_INDEXATION (IndexUnitActionPlugin.java)

- **Règle** : tâche consistant à indexer les métadonnées des unités archivistiques dans les bases internes de la solution logicielle Vitam, c'est-à-dire le titre des unités, leurs descriptions, leurs dates extrêmes, etc.
- **Type** : bloquant
- **Statuts** :
  - OK : les métadonnées des unités archivistiques ont été indexées avec succès (UNIT_METADATA_INDEXATION.OK = Succès de l'indexation des métadonnées de l'unité archivistique)
  - KO : au moins une des métadonnées des unités archivistiques n'a pas été indexée (UNIT_METADATA_INDEXATION.KO = Échec de l'indexation des métadonnées de l'unité archivistique)
  - FATAL : une erreur technique est survenue lors de l'indexation des métadonnées des unités archivistiques (UNIT_METADATA_INDEXATION.FATAL = Erreur technique lors de l'indexation des métadonnées de l'unité archivistique)

4.1.9.Processus d'enregistrement et écriture des métadonnées des objets et groupes d'objets(STP_OG_STORING)

4.1.9.1. Enregistrement des journaux du cycle de vie des groupes d'objets COMMIT_LIFE_CYCLE_OBJECT_GROUP (CommitLifeCycleObjectGroupActionHandler.java)

- **Règle** : tâche consistant à sécuriser en base les journaux du cycle de vie des groupes d'objets. Avant cette étape, les journaux du cycle de vie des groupes d'objets sont dans une collection temporaire afin de garder une cohérence entre les métadonnées indexées et les journaux lors d'une entrée en succès ou en échec, il n'y a pas d'évènement créée dans le journal du cycle de vie.
- **Type** : bloquant
- **Statuts** :
  - OK : la sécurisation des journaux du cycle de vie s'est correctement déroulée (COMMIT_LIFE_CYCLE_OBJECT_GROUP.OK = Succès de l'enregistrement des journaux du cycle de vie des groupes d'objets)
  - FATAL : une erreur technique est survenue lors de la sécurisation du journal du cycle de vie (COMMIT_LIFE_CYCLE_OBJECT_GROUP.FATAL = Erreur technique lors de l'enregistrement des journaux du cycle de vie des groupes d'objets)

4.1.9.2. Écriture des métadonnées du groupe d'objets et objets sur l'offre de stockage OG_METADATA_STORAGE (StoreMetaDataObjectGroupActionPlugin)

- **Règle** : tâche consistant à sauvegarder les métadonnées liées aux groupes d'objets ainsi que leurs journaux de cycle de vie sur les offres de stockage en fonction de la stratégie de stockage
- **Type** : bloquant
- **Statuts** :
  - OK : les métadonnées des groupes d'objets et objets ont été sauvegardées avec succès (OG_METADATA_STORAGE.OK = Succès de l'écriture des métadonnées des objets et groupes d'objets sur l'offre de stockage)
  - KO : les métadonnées des groupes d'objets et objets n'ont pas été sauvegardées (OG_METADATA_STORAGE.KO = Échec de l'écriture des métadonnées des objets et groupes

d'objets sur l'offre de stockage)

- FATAL : une erreur technique est survenue lors de l'écriture des métadonnées du groupe d'objets sur les offres de stockage (OG_METADATA_STORAGE.FATAL = Erreur technique lors de l'écriture des métadonnées du groupe d'objets sur les offres de stockage)

4.1.10.Processus d'enregistrement et écriture des unités archivistiques (STP_UNIT_STORING)

4.1.10.1. Enregistrement du journal du cycle de vie des unités archivistiques COMMIT_LIFE_CYCLE_UNIT (AccessInternalModuleImpl.java)

- **Règle** : tâche consistant à sécuriser en base les journaux du cycle de vie des unités archivistiques. Avant cette étape, les journaux du cycle de vie des unités archivistiques sont dans une collection temporaire afin de garder une cohérence entre les métadonnées indexées et les journaux lors d'une entrée en succès ou en échec.
- **Type** : bloquant
- **Statuts** :
  - OK : la sécurisation des journaux du cycle de vie s'est correctement déroulée (COMMIT_LIFE_CYCLE_UNIT.OK = Succès de l'enregistrement des journaux du cycle de vie des unités archivistiques)
  - FATAL : une erreur technique est survenue lors de la sécurisation des journaux du cycle de vie (COMMIT_LIFE_CYCLE_UNIT.FATAL = Erreur technique lors de l'enregistrement des journaux du cycle de vie des unités archivistiques)

4.1.10.2. Écriture des métadonnées de l'unité archivistique sur l'offre de stockage UNIT_METADATA_STORAGE (AccessInternalModuleImpl.java)

- **Règle** : tâche consistant à sauvegarder les métadonnées et des journaux de cycle de vie des unités archivistiques sur les offres de stockage en fonction de la stratégie de stockage. Pas d'évènements stockés dans le journal de cycle de vie
- **Type** : bloquant
- **Statuts** :
  - OK : l'écriture des métadonnées de l'unité archivistique sur les offres de stockage s'est correctement déroulée (UNIT_METADATA_STORAGE.OK = Succès de l'écriture des métadonnées de l'unité archivistique sur les offres de stockage)
  - KO : l'écriture des métadonnées de l'unité archivistique sur les offres de stockage n'a pas été effectuée (UNIT_METADATA_STORAGE.KO = Échec de l'écriture des métadonnées de l'unité archivistique sur les offres de stockage)
  - FATAL : une erreur technique est survenue lors de la sécurisation du journal du cycle de vie (UNIT_METADATA_STORAGE.FATAL = Erreur technique lors de l'écriture des métadonnées de l'unité archivistique sur les offres de stockage)

4.1.11.Processus de mise à jour des groupes d'objets (STP_UPDATE_OBJECT_GROUP)

4.1.11.1. Mise à jour des groupes d'objets OBJECT_GROUP_UPDATE (AccessInternalModuleImpl.java)

- **Règle** : Tâche consistant à mettre à jour les groupes d'objet techniques
- **Type** : bloquant

- OK=Succès de la mise à jour des groupes d''objets existants
- KO=Échec lors de la mise à jour des groupes d'objets existants
- FATAL=Erreur fatale lors de la mise à jour des groupes d'objets existants
- WARNING=Avertissement lors de la mise à jour des groupes d'objets existants

4.1.11.2. Enregistrement du journal du cycle de vie des groupes d'objets COMMIT_LIFE_CYCLE_OBJECT_GROUP (CommitLifeCycleObjectGroupActionHandler.java)

- **Règle** : tâche consistant à sécuriser en base les journaux du cycle de vie des groupes d'objets. Avant cette étape, les journaux du cycle de vie des groupes d'objets sont dans une collection temporaire afin de garder une cohérence entre les métadonnées indexées et les journaux lors d'une entrée en succès ou en échec, et il n'y a pas d'évènement créée dans le journal du cycle de vie.
- **Type** : bloquant
- **Statuts** :
  - OK : la sécurisation des journaux du cycle de vie s'est correctement déroulée (COMMIT_LIFE_CYCLE_OBJECT_GROUP.OK = Succès de l'enregistrement des journaux du cycle de vie des groupes d'objets)
  - FATAL : une erreur technique est survenue lors de la sécurisation du journal du cycle de vie (COMMIT_LIFE_CYCLE_OBJECT_GROUP.FATAL = Erreur technique lors de l'enregistrement des journaux du cycle de vie des groupes d'objets)

4.1.11.3. Écriture des métadonnées des groupes d'objets sur l'offre de stockage OG_METADATA_STORAGE (StoreMetaDataObjectGroupActionPlugin)

- **Règle** : tâche consistant à sauvegarder les métadonnées liées aux groupes d'objets ainsi que leurs journaux de cycle de vie sur les offres de stockage en fonction de la stratégie de stockage
- **Type** : bloquant
- **Statuts** :
  - OK : les métadonnées des groupes d'objets et objets ont été sauvegardées avec succès (OG_METADATA_STORAGE.OK = Succès de l'écriture des métadonnées des objets et groupes d'objets sur l'offre de stockage)
  - KO : les métadonnées des groupes d'objets et objets n'ont pas été sauvegardées (OG_METADATA_STORAGE.KO = Échec de l'écriture des métadonnées des objets et groupes d'objets sur l'offre de stockage)
  - FATAL : une erreur technique est survenue lors de l'écriture des métadonnées du groupe d'objets sur les offres de stockage (OG_METADATA_STORAGE.FATAL = Erreur technique lors de l'écriture des métadonnées du groupe d'objets sur les offres de stockage)

4.1.12.Processus d'alimentation du registre des fonds (STP_ACCESSION_REGISTRATION)

4.1.12.1. Alimentation du registre des fonds ACCESSION_REGISTRATION

- **Règle** : tâche consistant à enregistrer dans le registre des fonds des informations concernant la nouvelle entrée (nombre d'objets, volumétrie...). Ces informations viennent s'ajouter aux informations existantes pour un même service producteur. Si aucune information n'existait préalablement, alors un nouveau document est créé dans la base de données concernant ce producteur. Une fois cette action d'ajout ou de mise à jour effectuée, la solution logicielle Vitam calcule et enregistre une information agrégée de l'état des stocks du service producteur concerné (dans la collection AccessionRegisterDetail).

- **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 : une erreur technique est survenue lors de l'alimentation du registre des fonds (ACCESSION_REGISTRATION.FATAL = Erreur technique lors de l'alimentation du registre des fonds)

4.1.13.Processus de finalisation de l'entrée (STP_INGEST_FINALISATION)

4.1.13.1. Notification de la fin de l'opération d'entrée ATR_NOTIFICATION (TransferNotificationActionHandler.java)

- **Règle** : tâche consistant à générer la notification de réponse (ArchiveTransferReply ou ATR) une fois toutes les étapes passées, en succès, avertissement ou échec, puis écriture de cette notification dans l'offre de stockage et envoi au service versant
- **Type** : non bloquant
- **Statuts** :
  - OK : le message de réponse a été correctement généré, écrit sur l'offre de stockage et envoyé au service versant (ATR_NOTIFICATION.OK = Succès de la notification de la fin de l'opération d'entrée à l'opérateur de versement)
  - KO : le message de réponse n'a pas été correctement généré, écrit sur l'offre de stockage ou envoyé au service versant (ATR_NOTIFICATION.KO = Échec de la notification de la fin de l'opération d'entrée à l'opérateur de versement)
  - FATAL : une erreur technique est survenue lors de la notification de la fin de l'opération (ATR_NOTIFICATION.FATAL = Erreur technique lors de la notification de la fin de l'opération d'entrée à l'opérateur de versement)

4.1.13.2. Mise en cohérence des journaux du cycle de vie ROLL_BACK (RollBackActionHandler.java)

- **Règle** : Purge des collections temporaires des journaux du cycle de vie
- **Type** : bloquant
- **Statuts** :
  - OK : la purge s'est correctement déroulée (ROLL_BACK.OK = Succès de la mise en cohérence des journaux du cycle de vie)
  - FATAL : une erreur technique est survenue lors de la purge (ROLL_BACK.FATAL = Erreur technique lors de la mise en cohérence des journaux du cycle de vie)

4.1.14.Le cas du processus d'entrée « test à blanc »

Il est possible de procéder à un versement dit « à blanc », pour tester la conformité du SIP par rapport à la forme attendue par la solution logicielle Vitam sans pour autant le prendre en charge. Dans ce cas, le processus d'entrée à blanc diffère du processus d'entrée « classique » en ignorant un certain nombre d'étapes.

Les étapes non exécutées dans le processus d'entrée à blanc sont les suivantes :

- Écriture et indexation des objets et groupes d'objets (STP_OBJ_STORING)
- Indexation des unités archivistiques (STP_UNIT_METADATA)

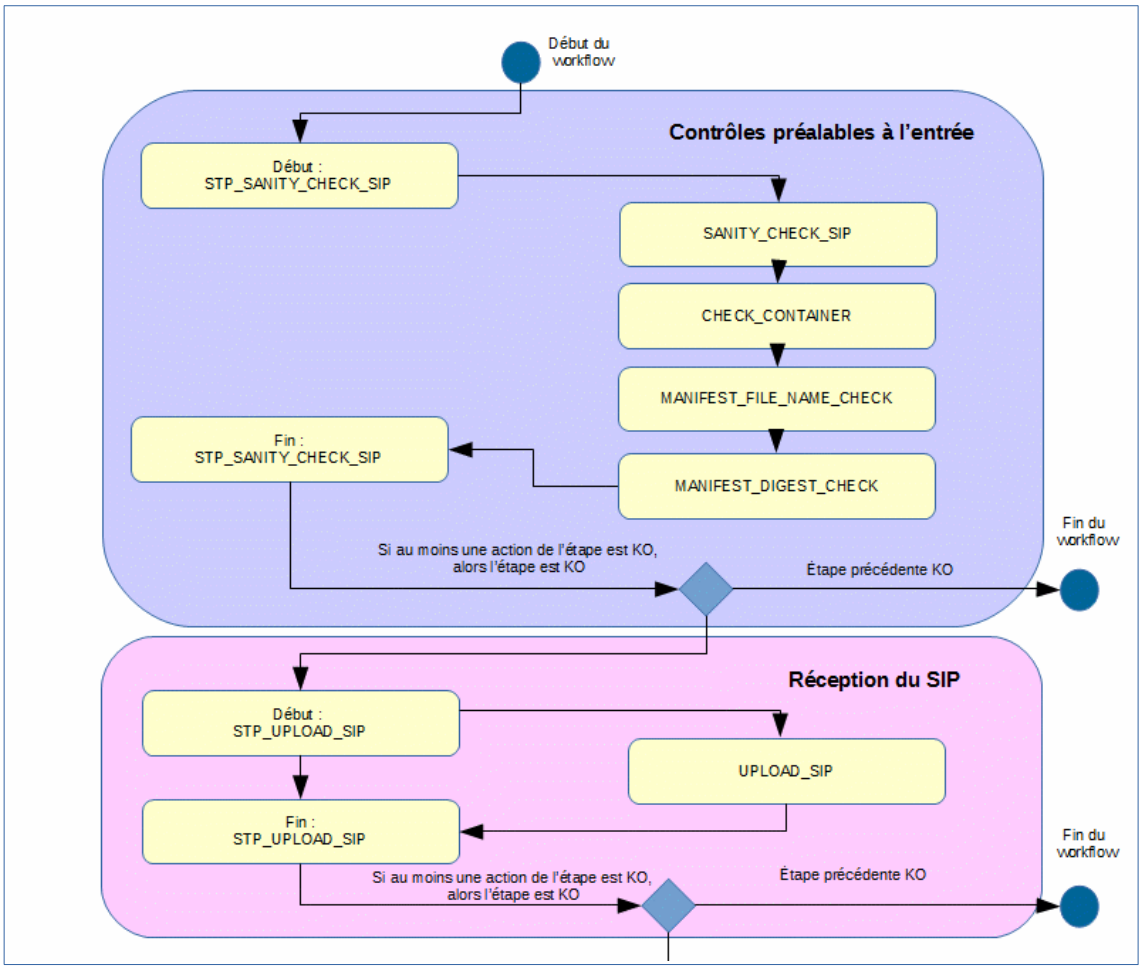
- Enregistrement et écriture des métadonnées des objets et groupes d'objets (STP_OG_STORING)
- Enregistrement et écriture des unités archivistiques (STP_UNIT_STORING)
- Rangement des métadonnées des objets (STP_UPDATE_OBJECT_GROUP)
- Alimentation du registre des fonds (STP_ACCESSION_REGISTRATION)

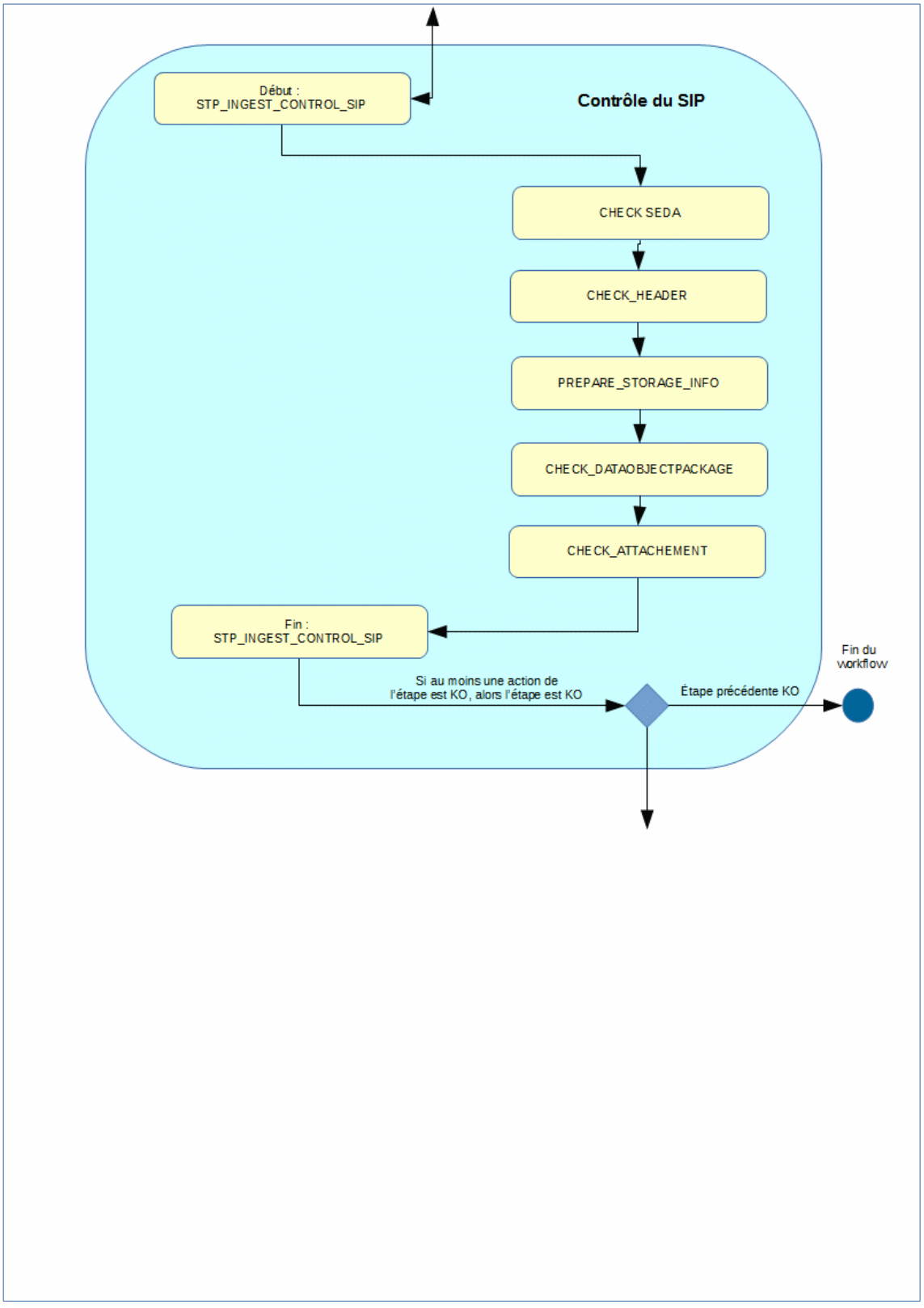
  Les tâches et traitements relatifs à toutes ces étapes sont donc également ignorés.

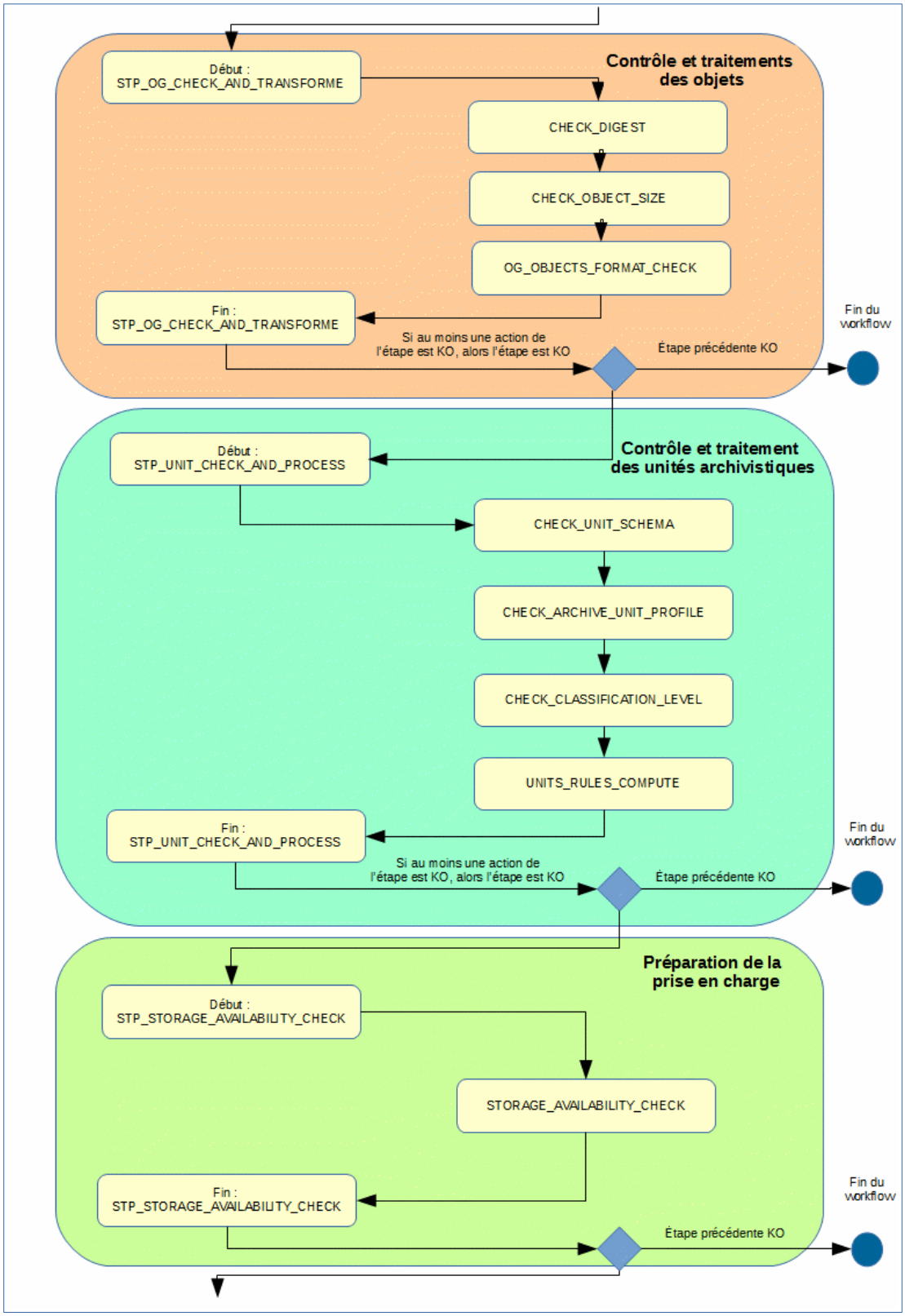
4.1.15.Structure du Workflow de l'entrée

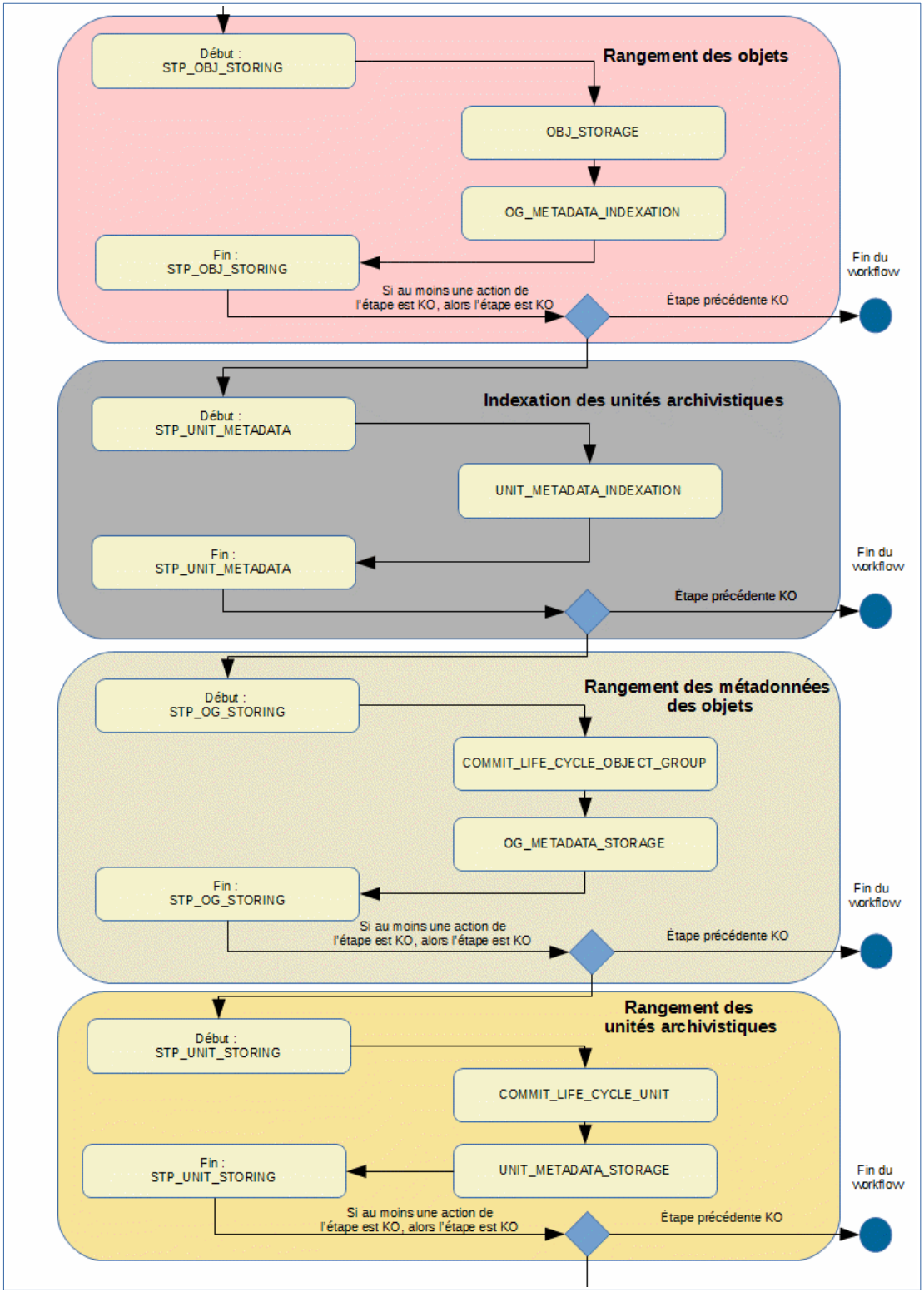
Le workflow mis en place dans la solution logicielle Vitam est défini dans l'unique fichier *« DefaultIngestWorkflow.json »*. Ce fichier est disponible dans /sources/processing/processing-management/src/main/resources/workflows. Il décrit le processus d'entrée (hors Ingest externe) pour entrer un SIP, indexer les métadonnées et stocker les objets contenus dans le SIP.

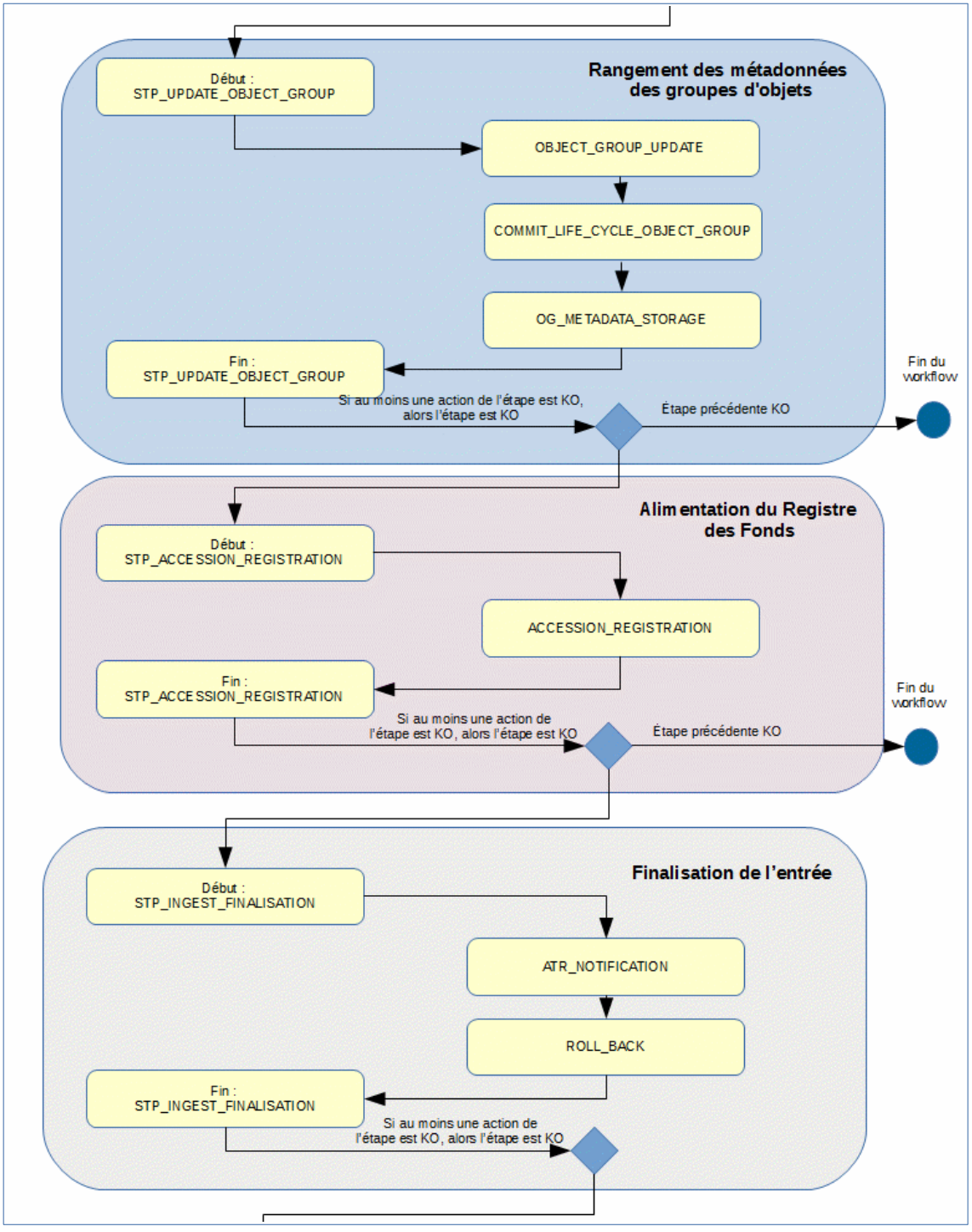
D'une façon synthétique, le workflow est décrit de cette façon :




Début :
STP_INGEST_CONTROL_SIP
Contrôle du SIP
CHECK SEDA
CHECK_HEADER
PREPARE_STORAGE_INFO
CHECK_DATAOBJECTPACKAGE
CHECK_ATTACHEMENT
Fin :
STP_INGEST_CONTROL_SIP
Si au moins une action de
l'étape est KO, alors l'étape est KO
Étape précédente KO
Fin du
workflow


Contrôle et traitements des objets
Début : STP_OG_CHECK_AND_TRANSFORME
CHECK_DIGEST
CHECK_OBJECT_SIZE
OG_OBJECTS_FORMAT_CHECK
Fin : STP_OG_CHECK_AND_TRANSFORME
Si au moins une action de l'étape est KO, alors l'étape est KO
Étape précédente KO
Fin du workflow
Contrôle et traitement des unités archivistiques
Début : STP_UNIT_CHECK_AND_PROCESS
CHECK_UNIT_SCHEMA
CHECK_ARCHIVE_UNIT_PROFILE
CHECK_CLASSIFICATION_LEVEL
UNITS_RULES_COMPUTE
Fin : STP_UNIT_CHECK_AND_PROCESS
Si au moins une action de l'étape est KO, alors l'étape est KO
Étape précédente KO
Fin du workflow
Préparation de la prise en charge
Début : STP_STORAGE_AVAILABILITY_CHECK
STORAGE_AVAILABILITY_CHECK
Fin : STP_STORAGE_AVAILABILITY_CHECK
Étape précédente KO
Fin du workflow


Début :
STP_OBJ_STORING
Rangement des objets
OBJ_STORAGE
OG_METADATA_INDEXATION
Fin :
STP_OBJ_STORING
Si au moins une action de
l'étape est KO, alors l'étape est KO
Étape précédente KO
Fin du
workflow
Début :
STP_UNIT_METADATA
Indexation des unités archivistiques
UNIT_METADATA_INDEXATION
Fin :
STP_UNIT_METADATA
Étape précédente KO
Fin du
workflow
Début :
STP_OG_STORING
Rangement des métadonnées
des objets
COMMIT_LIFE_CYCLE_OBJECT_GROUP
OG_METADATA_STORAGE
Fin :
STP_OG_STORING
Si au moins une action de
l'étape est KO, alors l'étape est KO
Étape précédente KO
Fin du
workflow
Début :
STP_UNIT_STORING
Rangement des
unités archivistiques
COMMIT_LIFE_CYCLE_UNIT
Fin :
STP_UNIT_STORING
UNIT_METADATA_STORAGE
Si au moins une action de
l'étape est KO, alors l'étape est KO
Étape précédente KO
Fin du
workflow


Rangement des métadonnées des groupes d'objets
Début : STP_UPDATE_OBJECT_GROUP
OBJECT_GROUP_UPDATE
COMMIT_LIFE_CYCLE_OBJECT_GROUP
OG_METADATA_STORAGE
Fin : STP_UPDATE_OBJECT_GROUP
Si au moins une action de l'étape est KO, alors l'étape est KO
Étape précédente KO
Fin du workflow
Alimentation du Registre des Fonds
Début : STP_ACCESSION_REGISTRATION
ACCESSION_REGISTRATION
Fin : STP_ACCESSION_REGISTRATION
Si au moins une action de l'étape est KO, alors l'étape est KO
Étape précédente KO
Fin du workflow
Finalisation de l'entrée
Début : STP_INGEST_FINALISATION
ATR_NOTIFICATION
ROLL_BACK
Fin : STP_INGEST_FINALISATION
Si au moins une action de l'étape est KO, alors l'étape est KO

4.2.Workflow d'entrée d'un plan de classement

Cette section décrit le processus d'entrée d'un plan de classement dans la solution logicielle Vitam.

Le processus d'entrée d'un plan de classement est identique au workflow d'entrée d'un SIP. Il débute lors du lancement du téléchargement d'un plan de classement dans la solution logicielle Vitam. Toutes les étapes, tâches et traitements sont journalisés dans le journal des opérations.

Les étapes, tâches et traitements associées ci-dessous décrivent le processus d'entrée d'un plan (clé et description de la clé associée dans le journal des opérations), non encore abordées dans la description de l'entrée d'un SIP.

Toutes les étapes, tâches et traitements sont journalisées dans le journal des opérations. Les étapes et actions associées ci-dessous décrivent le processus (clé et description de la clé associée dans le journal des opérations) tel qu'implémenté dans la version actuelle de la solution logicielle Vitam.

4.2.1.Processus d'entrée d'un plan de classement (vision métier)

La structure d'un plan de classement diffère de celle d'un SIP par l'absence d'objet et de vérification de l'offre de stockage. Il s'agit plus simplement d'une arborescence représentée par des unités archivistiques. Ce processus partage donc certaines étapes avec celui du transfert d'un SIP classique, en ignore certaines et rajoute des tâches additionnelles.

4.2.1.1. Traitement additionnel dans la tâche CHECK_DATAOBJECTPACKAGE

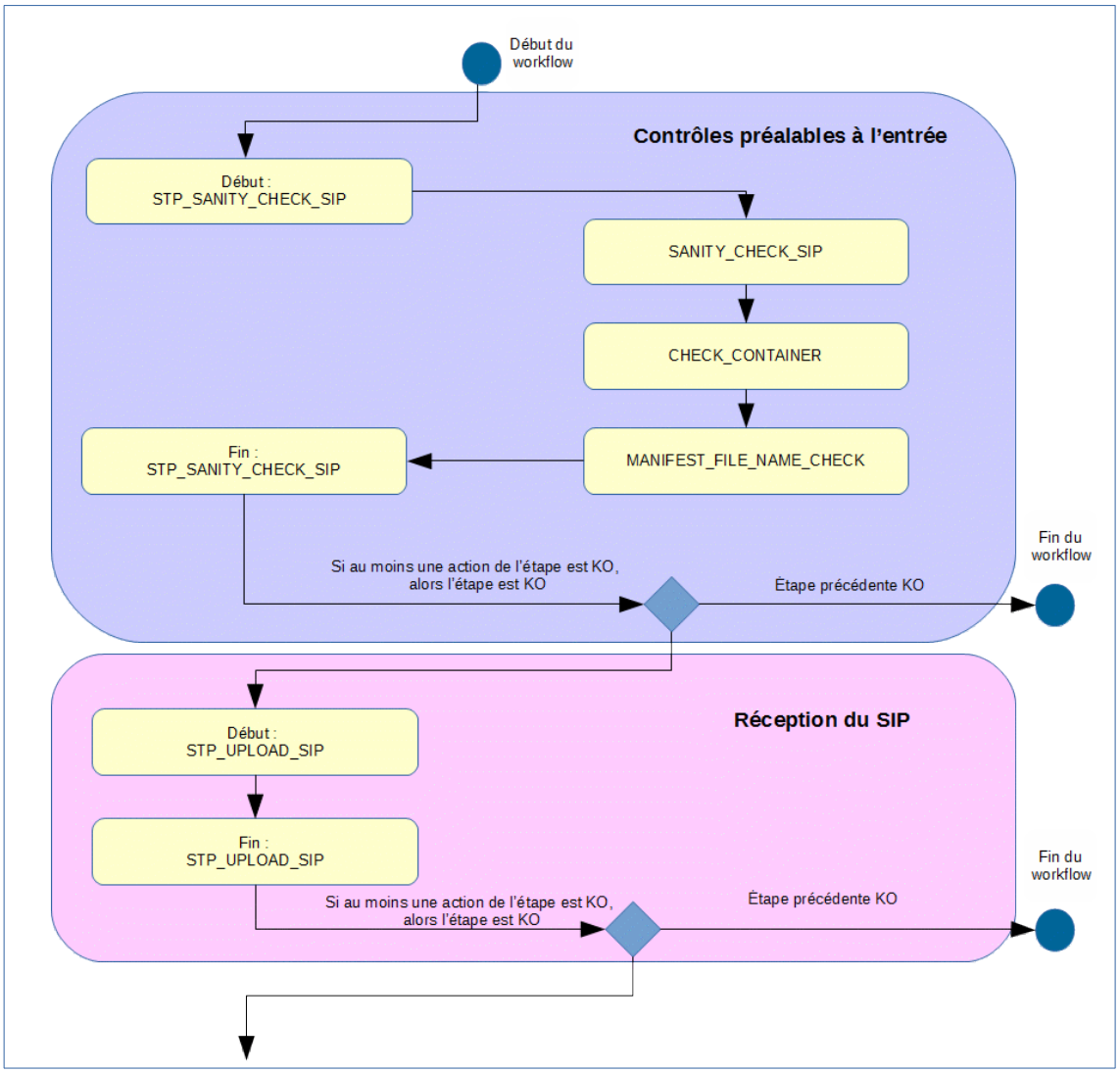
Vérification de la non-existence d'objets (CHECK_NO_OBJECT)

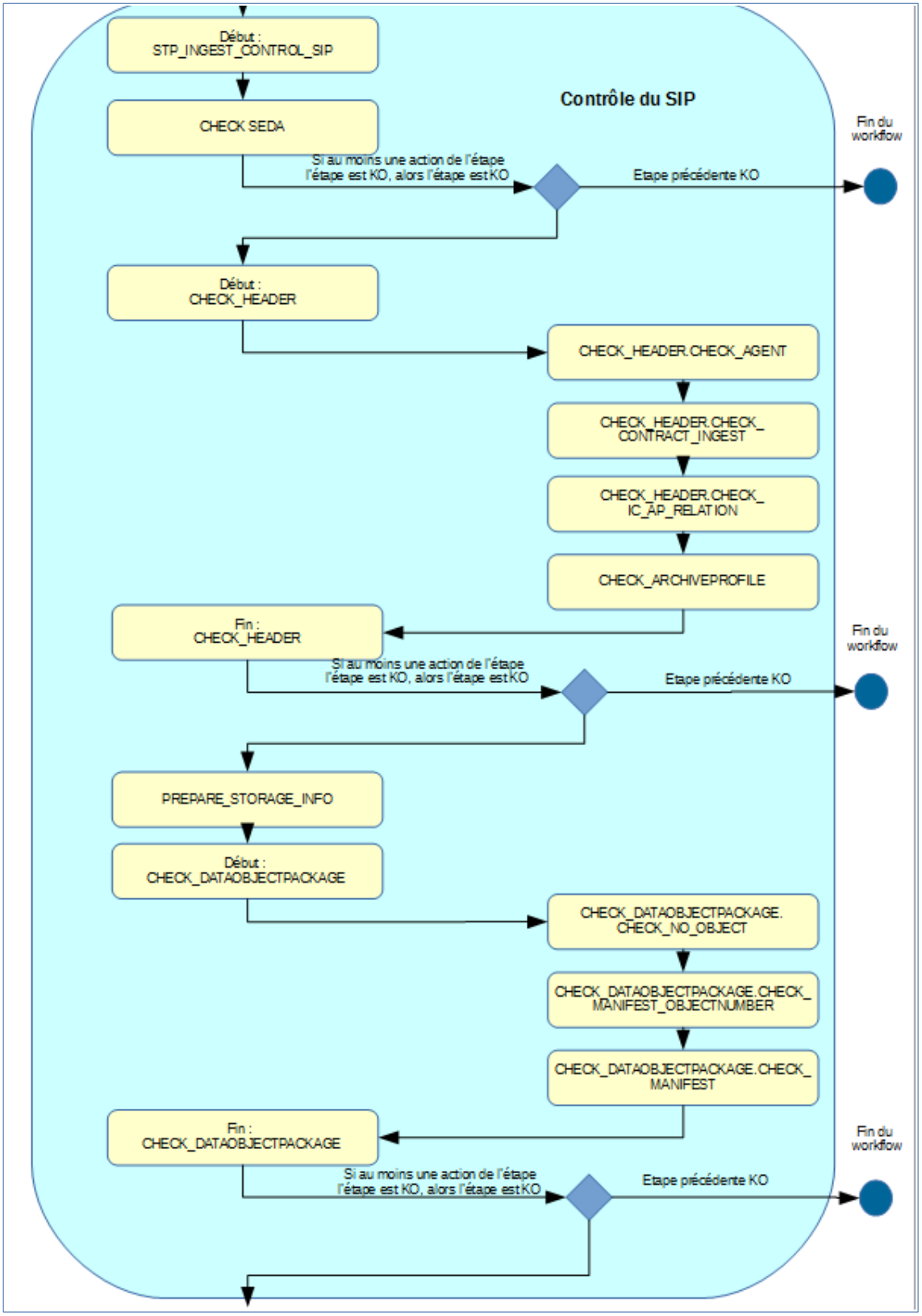
- **Règle** : traitement consistant à vérifier qu'il n'y a pas d'objet binaire dans le bordereau de transfert du plan de classement
- **Type** : bloquant
- **Statuts** :
  - OK : aucun objet binaire n'est présent dans le bordereau de transfert (CHECK_DATAOBJECTPACKAGE.CHECK_NO_OBJECT.OK = Succès de la vérification de l'absence d'objet)
  - KO : des objets binaires sont présents dans le bordereau de transfert (CHECK_DATAOBJECTPACKAGE.CHECK_NO_OBJECT.KO = Échec de la vérification de l'absence d'objet : objet(s) trouvé(s))
  - FATAL : une erreur technique est survenue lors de la vérification de la non-existence d'objet binaire (CHECK_DATAOBJECTPACKAGE.CHECK_NO_OBJECT.FATAL = Erreur technique lors de la vérification de l'absence d'objet)

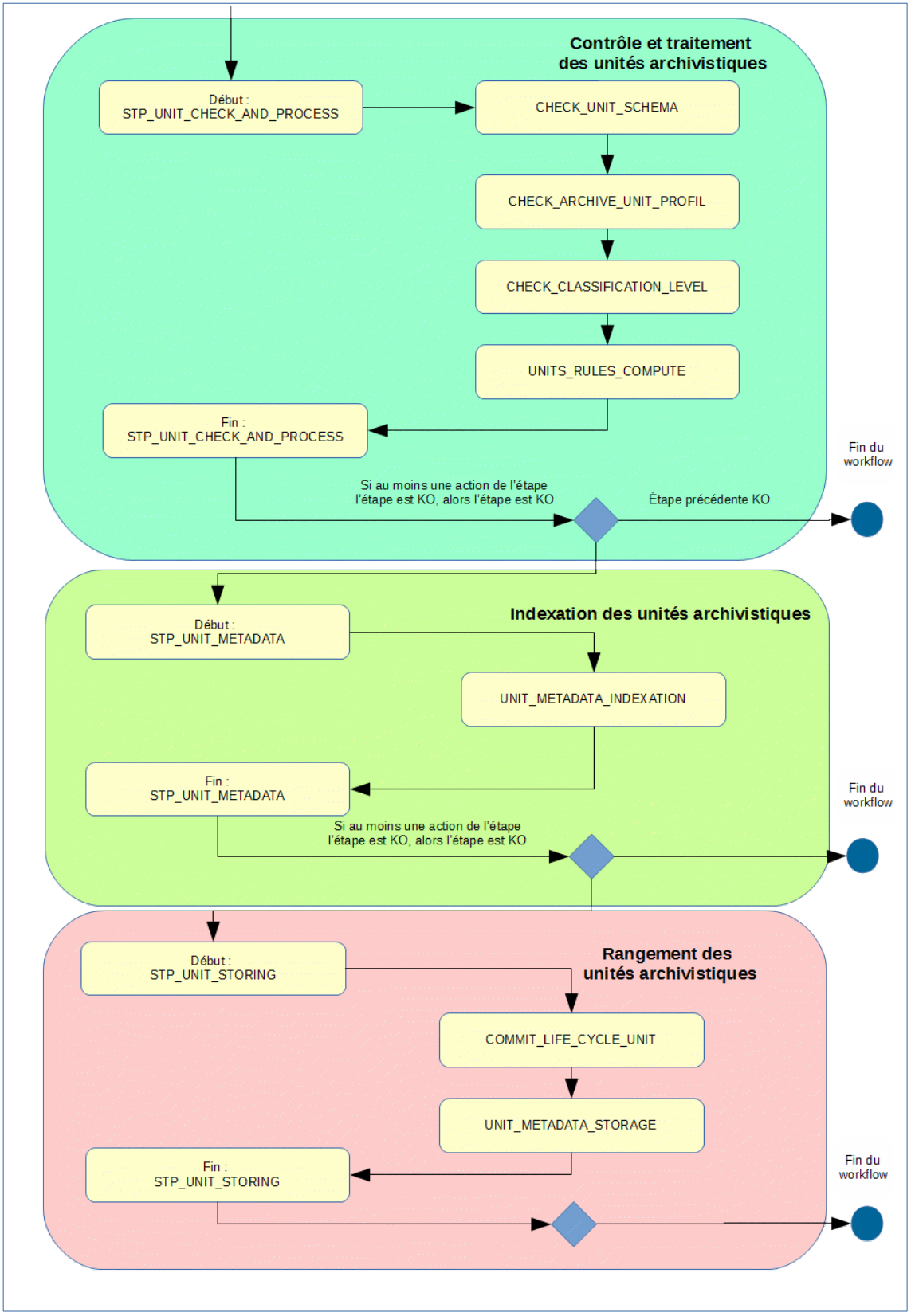
4.2.2.Structure de workflow d'entrée d'un plan de classement

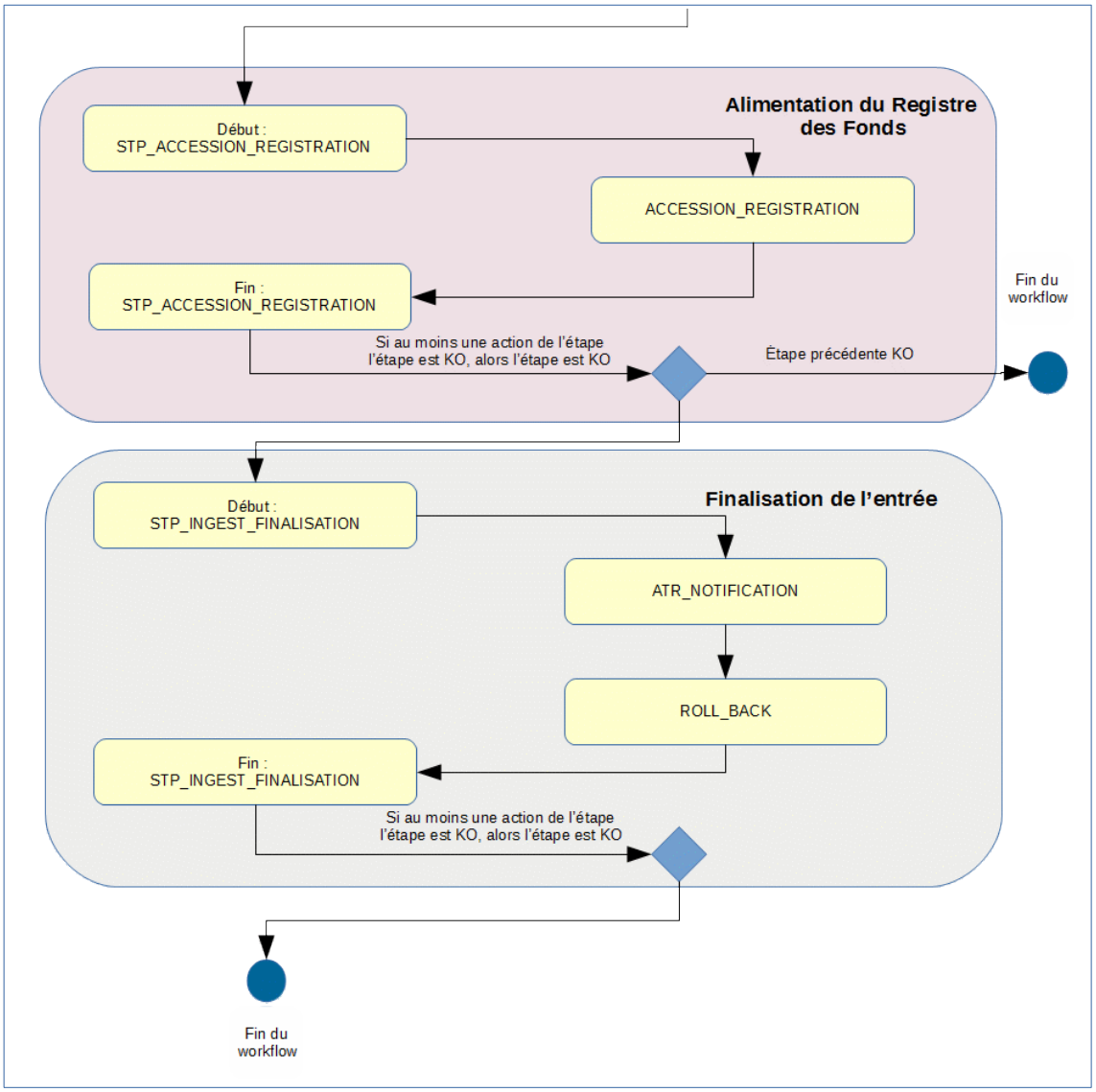
Le workflow actuel mis en place dans la solution logicielle Vitam est défini dans le fichier « *DefaultFilingSchemeWorkflow.json* ». Ce fichier est disponible dans : sources/processing/processing-management/src/main/resources/workflows.

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




Début :
STP_INGEST_CONTROL_SIP
Contrôle du SIP
CHECK SEDA
Fin du workflow
Si au moins une action de l'étape l'étape est KO, alors l'étape est KO
Etape précédente KO
Début :
CHECK_HEADER
CHECK_HEADER.CHECK_AGENT
CHECK_HEADER.CHECK_ CONTRACT_INGEST
CHECK_HEADER.CHECK_ IC_AP_RELATION
CHECK_ARCHIVEPROFILE
Fin :
CHECK_HEADER
Fin du workflow
Si au moins une action de l'étape l'étape est KO, alors l'étape est KO
Etape précédente KO
PREPARE_STORAGE_INFO
Début :
CHECK_DATAOBJECTPACKAGE
CHECK_DATAOBJECTPACKAGE. CHECK_NO_OBJECT
CHECK_DATAOBJECTPACKAGE.CHECK_ MANIFEST_OBJECTNUMBER
CHECK_DATAOBJECTPACKAGE.CHECK_ MANIFEST
Fin :
CHECK_DATAOBJECTPACKAGE
Fin du workflow
Si au moins une action de l'étape l'étape est KO, alors l'étape est KO
Etape précédente KO


Contrôle et traitement
des unités archivistiques
Début :
STP_UNIT_CHECK_AND_PROCESS
CHECK_UNIT_SCHEMA
CHECK_ARCHIVE_UNIT_PROFIL
CHECK_CLASSIFICATION_LEVEL
UNITS_RULES_COMPUTE
Fin :
STP_UNIT_CHECK_AND_PROCESS
Si au moins une action de l'étape
l'étape est KO, alors l'étape est KO
Étape précédente KO
Fin du
workflow
Indexation des unités archivistiques
Début :
STP_UNIT_METADATA
UNIT_METADATA_INDEXATION
Fin :
STP_UNIT_METADATA
Si au moins une action de l'étape
l'étape est KO, alors l'étape est KO
Fin du
workflow
Rangement des
unités archivistiques
Début :
STP_UNIT_STORING
COMMIT_LIFE_CYCLE_UNIT
UNIT_METADATA_STORAGE
Fin :
STP_UNIT_STORING
Fin du
workflow


Alimentation du Registre des Fonds
Début : STP_ACCESSION_REGISTRATION
ACCESSION_REGISTRATION
Fin : STP_ACCESSION_REGISTRATION
Si au moins une action de l'étape l'étape est KO, alors l'étape est KO
Étape précédente KO
Fin du workflow
Finalisation de l'entrée
Début : STP_INGEST_FINALISATION
ATR_NOTIFICATION
ROLL_BACK
Fin : STP_INGEST_FINALISATION
Si au moins une action de l'étape l'étape est KO, alors l'étape est KO
Fin du workflow

5. MASTERDATA

Cette section décrit les processus (workflows) d'administration des différents référentiels de la solution logicielle Vitam. Ceux-ci se construisent sur la base de fichiers à importer. La structure de ces fichiers et la description de leurs contenus sont décrites dans la documentation relative au modèle de données. Si un des fichiers importés contient des balises HTML, son contenu sera considéré comme dangereux et l'import sera rejeté. Ce rejet ne fera pas l'objet d'une opération et ne sera donc pas enregistré dans le journal des opérations. En revanche, une alerte de sécurité sera émise dans un log de sécurité de la solution logicielle Vitam, pour informer l'administrateur de cette tentative.

5.1. Workflow d'import d'un arbre de positionnement

Cette section décrit le processus permettant d'importer un arbre de positionnement dans la solution logicielle Vitam. La structure d'un arbre de positionnement diffère de celle d'un SIP en plusieurs points.

Un arbre ne doit pas avoir d'objet, ni de service producteur, ni de règle de gestion associée. Il s'agit plus simplement d'une arborescence représentée par des unités archivistiques. Ce processus partage donc certaines étapes avec celui du transfert d'un SIP classique, en ignore certaines et rajoute des tâches additionnelles.

Toutes les étapes, tâches et traitements sont journalisés dans le journal des opérations et décrivent le processus (clé et description de la clé associée dans le journal des opérations) tel qu'implémenté dans la version actuelle de la solution logicielle Vitam.

5.1.1. Processus d'import d'un arbre (HOLDINGSCHEME - vision métier)

Le processus d'import d'un arbre est identique au workflow d'entrée d'un SIP. Il débute lors du lancement du téléchargement de l'arbre dans la solution logicielle Vitam. Par ailleurs, toutes les étapes, tâches et traitements sont journalisés dans le journal des opérations.

La fin du processus peut prendre plusieurs statuts :

- **Statuts** :
  - OK : l'arbre de positionnement a été importé (HOLDINGSCHEME.OK = Succès de l'import de l'arbre de positionnement)
  - KO : l'arbre de positionnement n'a pas été importé (HOLDINGSCHEME.KO = Échec de l'import de l'arbre de positionnement)
  - FATAL : une erreur technique est survenue lors de l'import de l'arbre de positionnement (HOLDINGSCHEME.FATAL = Erreur technique lors de l'import de l'arbre de positionnement)

  Les étapes, tâches et traitements associées ci-dessous décrivent le processus d'import d'un arbre de positionnement (clé et description de la clé associée dans le journal des opérations), non encore abordées dans la description de l'entrée d'un SIP.

Traitement additionnel dans la tâche CHECK_DATAOBJECTPACKAGE (CheckDataObjectPackageActionHandler.java)

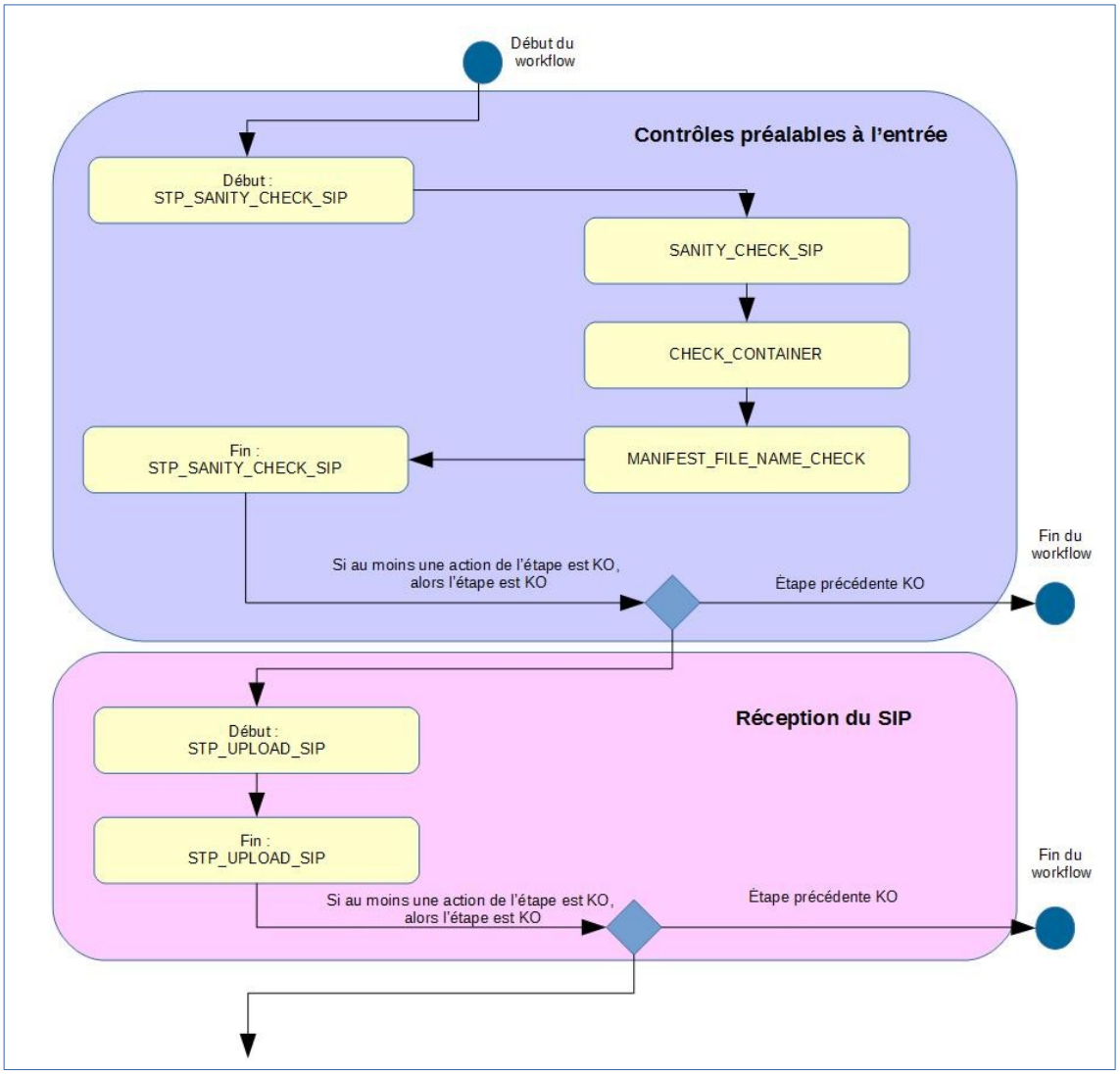
Vérification de la non-existence d'objets CHECK_NO_OBJECT (CheckDataObjectPackageActionHandler)

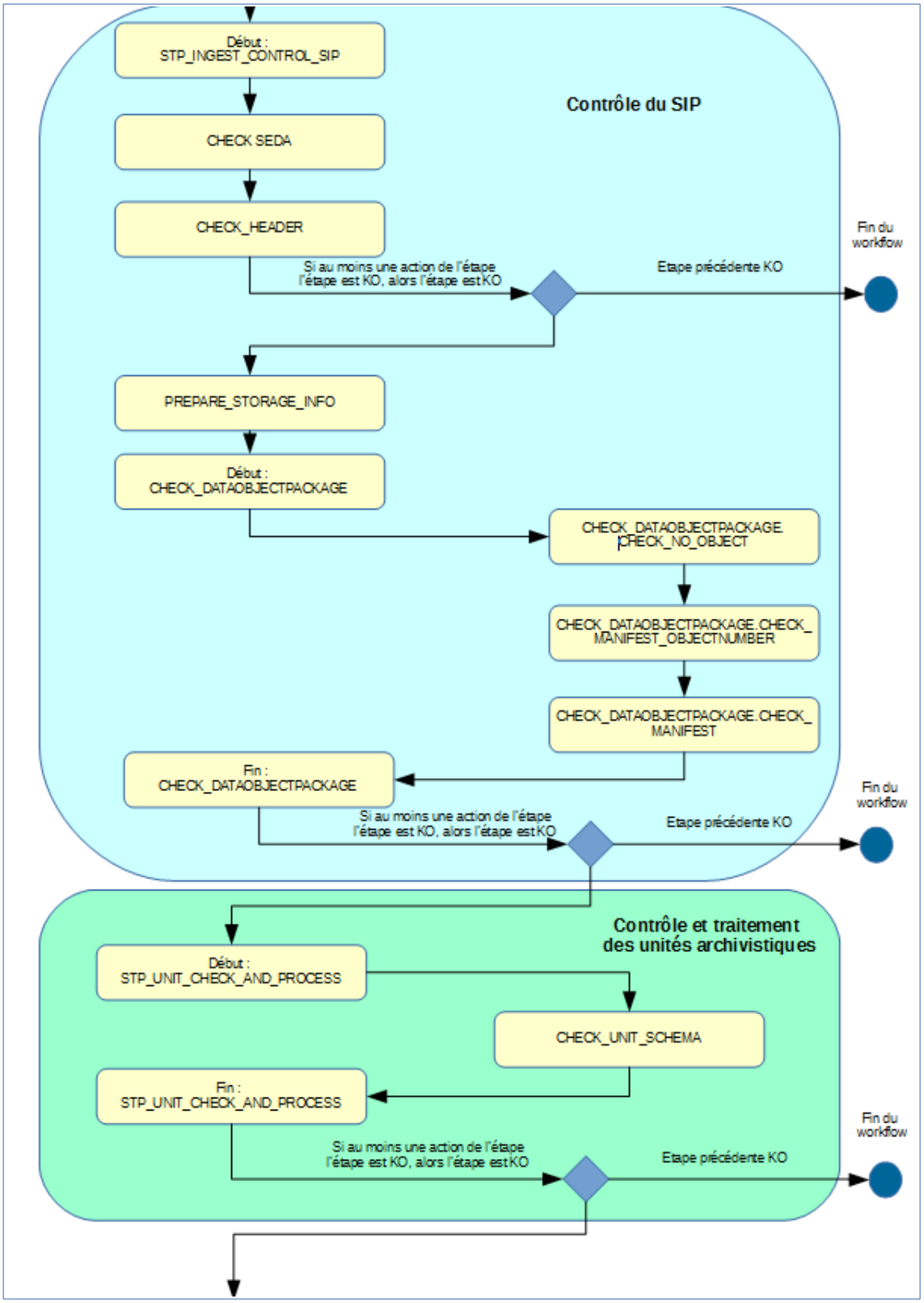
- **Règle** : traitement consistant à vérifier qu'il n'y a pas d'objet binaire dans le bordereau de transfert de l'arbre de positionnement.
  - **Statuts** :
    - OK : aucun objet binaire n'est présent dans le bordereau de transfert (CHECK_DATAOBJECTPACKAGE.CHECK_NO_OBJECT.OK = Succès de la vérification de l'absence d'objet)
    - KO : des objets binaires sont présents dans le bordereau de transfert (CHECK_DATAOBJECTPACKAGE.CHECK_NO_OBJECT.KO = Échec de la vérification de l'absence d'objet : objet(s) trouvé(s))

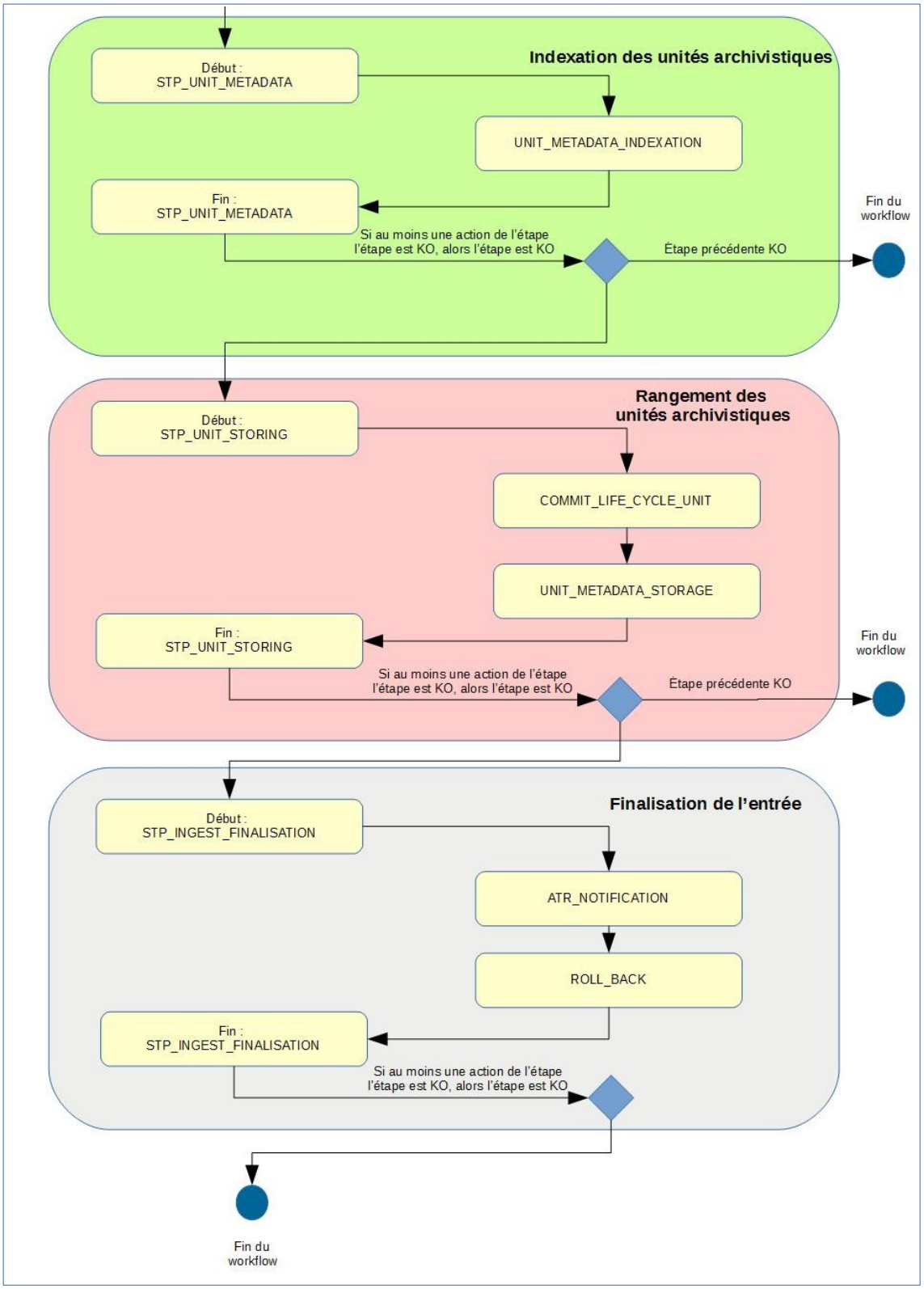
- FATAL : une erreur technique est survenue lors de la vérification de la non-existence d'objet binaire (CHECK_DATAOBJECTPACKAGE.CHECK_NO_OBJECT.FATAL = Erreur technique lors de la vérification de l'absence d'objet)

5.1.2.Structure du Workflow d'import d'un arbre de positionnement

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




Début :
STP_INGEST_CONTROL_SIP
Contrôle du SIP
CHECK SEDA
CHECK_HEADER
Si au moins une action de l'étape
l'étape est KO, alors l'étape est KO
Etape précédente KO
Fin du
workflow
PREPARE_STORAGE_INFO
Début :
CHECK_DATAOBJECTPACKAGE
CHECK_DATAOBJECTPACKAGE.
CHECK_NO_OBJECT
CHECK_DATAOBJECTPACKAGE.CHECK_
MANIFEST_OBJECTNUMBER
CHECK_DATAOBJECTPACKAGE.CHECK_
MANIFEST
Fin :
CHECK_DATAOBJECTPACKAGE
Si au moins une action de l'étape
l'étape est KO, alors l'étape est KO
Etape précédente KO
Fin du
workflow
Contrôle et traitement
des unités archivistiques
Début :
STP_UNIT_CHECK_AND_PROCESS
CHECK_UNIT_SCHEMA
Fin :
STP_UNIT_CHECK_AND_PROCESS
Si au moins une action de l'étape
l'étape est KO, alors l'étape est KO
Etape précédente KO
Fin du
workflow


Indexation des unités archivistiques
Début :
STP_UNIT_METADATA
UNIT_METADATA_INDEXATION
Fin :
STP_UNIT_METADATA
Si au moins une action de l'étape
l'étape est KO, alors l'étape est KO
Étape précédente KO
Fin du
workflow
Rangement des
unités archivistiques
Début :
STP_UNIT_STORING
COMMIT_LIFE_CYCLE_UNIT
UNIT_METADATA_STORAGE
Fin :
STP_UNIT_STORING
Si au moins une action de l'étape
l'étape est KO, alors l'étape est KO
Étape précédente KO
Fin du
workflow
Finalisation de l'entrée
Début :
STP_INGEST_FINALISATION
ATR_NOTIFICATION
ROLL_BACK
Fin :
STP_INGEST_FINALISATION
Si au moins une action de l'étape
l'étape est KO, alors l'étape est KO
Fin du
workflow

5.2.Workflow d'administration d'un référentiel de règles de gestion

Cette section décrit le processus (workflow) permettant d'importer et de mettre à jour un référentiel de règles de gestion dans la solution logicielle Vitam.

Toutes les étapes, tâches et traitements sont journalisés dans le journal des opérations et décrivent le processus (clé et description de la clé associée dans le journal des opérations) tel qu'implémenté dans la version actuelle de la solution logicielle Vitam.

5.2.1.Processus d'administration d'un référentiel de règles de gestion (STP_IMPORT_RULES)

L'import d'un référentiel de règles de gestion permet de vérifier le formalisme de ce dernier, notamment que les données obligatoires sont bien présentes pour chacune des règles. Tous les éléments réalisés au cours de ce processus sont exécutés dans une seule étape. Cet import concerne aussi bien l'import initial (aucune règle de gestion pré-existante) que la mise à jour du référentiel.

Ce processus d'import débute lors du lancement du téléchargement dans la solution logicielle Vitam du fichier CSV contenant le référentiel.

La fin du processus peut prendre plusieurs statuts :

- **Statuts** :
  - OK : le référentiel des règles de gestion a été importé (STP_IMPORT_RULES.OK = Succès du processus d'import du référentiel des règles de gestion)
  - WARNING : le référentiel des règles de gestion a été importé et ce nouvel import modifie des règles de gestions préalablement utilisées par des unités archivistiques dans la solution logicielle Vitam (STP_IMPORT_RULES.WARNING = Avertissement lors du processus d'import des règles de gestion, des règles de gestions ont été modifiées et sont utilisées par des unités archivistiques existantes)
  - KO : le référentiel des règles de gestion n'a pas été importé (STP_IMPORT_RULES.KO = Échec du processus d'import du référentiel des règles de gestion)
  - FATAL : une erreur technique est survenue lors de l'import du référentiel des règles de gestion (STP_IMPORT_RULES.FATAL = Erreur technique lors du processus d'import du référentiel des règles de gestion)

5.2.1.1. Contrôle de la conformité du fichier des règles de gestion CHECK_RULES (UnitsRulesComputePlugin.java)

- **Règle** : tâche consistant à
  - contrôler qu'aucune règle supprimée du référentiel n'est utilisée par une unité archivistique déjà présente dans le système,
  - contrôler les règles modifiées utilisées par des unités archivistiques déjà présentes dans le système,
  - vérifier que les informations obligatoires minimales ont bien été remplies pour chacune des règles, conformément aux exigences du référentiel des règles de gestion. La liste de ces exigences est décrite dans le document modèle de données.

  De plus le fichier remplit les conditions suivantes :
  - il est au format CSV
  - les informations suivantes sont toutes décrites dans cet ordre
    - RuleId
    - RuleType
    - RuleValue
    - RuleDescription
    - RuleDuration

        - RuleMeasurement
    - Aucune règle supprimée n'est utilisée par une unité archivistique
    - Aucune opération d'import de référentiel de règle de gestion n'a lieu en même temps
    - Si le tenant définit des durées minimales pour des catégories de règles de gestion (configuration de sécurité), les règles de gestion importées doivent avoir des durées supérieures ou égales à ces durées minimales de sécurité
- **Type** : bloquant
- **Statuts** :
    - OK : les règles ci-dessus sont respectées (CHECK_RULES.OK = Succès du contrôle de la conformité du fichier des règles de gestion)
    - WARNING : une règle modifiée par l'import du référentiel est actuellement utilisée par une unité archivistique (CHECK_RULES.WARNING = Avertissement lors du contrôle de la conformité du fichier des règles de gestion)
    - KO :
        - Cas 1 : une des règles ci-dessus n'est pas respectée. Le détail des erreurs est inscrit dans le rapport d'import du référentiel (CHECK_RULES.KO = Échec du contrôle de la conformité du fichier des règles de gestion)
        - Cas 2 : le fichier CSV n'est pas reconnu comme un CSV valide (CHECK_RULES.INVALID_CSV.KO = Échec du contrôle de la conformité du fichier des règles de gestion : fichier CSV invalide)
        - Cas 3 : une opération de mise à jour du référentiel est déjà en cours (CHECK_RULES.IMPORT_IN_PROCESS.KO = Échec du contrôle de la conformité du fichier car une mise à jour du référentiel est déjà en cours
        - Cas 4 : au moins une règle de gestion a une durée qui est inférieure à la durée minimale requise sur ce tenant. Selon la configuration de la solution logicielle Vitam, ce cas peut provoquer une alerte de sécurité, enregistrée dans les logs de sécurité. (CHECK_RULES.MAX_DURATION_EXCEEDS.KO = Échec lors du contrôle de sécurité des règles de gestion. Les durées des règles de gestion doivent être supérieures ou égales aux durées minimales requises par le tenant)
    - FATAL : une erreur technique est survenue lors du contrôle des règles de gestion (CHECK_RULES.FATAL = Erreur technique lors du contrôle des règles de gestion)
- Dans le rapport de l'import du référentiel des règles de gestion, des clés plus détaillées peuvent y être inscrites, selon les erreurs rencontrées :
    - Le fichier importé n'est pas au format CSV (STP_IMPORT_RULES_NOT_CSV_FORMAT.KO = Le fichier importé n'est pas au format CSV)
    - Il existe plusieurs fois le même RuleId (STP_IMPORT_RULES_RULEID_DUPLICATION.KO = Il existe plusieurs fois le même RuleId. Ce RuleId doit être unique dans l'ensemble du référentiel)
    - Au moins une RuleType est incorrecte (STP_IMPORT_RULES_WRONG_RULETYPE_UNKNOW.KO = Au moins une RuleType est incorrecte. RuleType autorisés : AppraisalRule, AccessRule, StorageRule, DisseminationRule, ReuseRule, ClassificationRule)
    - Au moins une valeur obligatoire est manquante (STP_IMPORT_RULES_MISSING_INFORMATION.KO = Au moins une valeur obligatoire est manquante. Valeurs obligatoires : RuleID, RuleType, RuleValue, RuleDuration, RuleMeasurement)
    - Des valeurs de durée sont incorrectes pour RuleMeasurement (STP_IMPORT_RULES_WRONG_RULEMEASUREMENT.KO = Au moins un champ RuleDuration a une valeur incorrecte. La valeur doit être un entier positif ou nul, ou être indiquée unlimited)
    - Au moins un champ RuleDuration a une valeur incorrecte (STP_IMPORT_RULES_WRONG_RULEDURATION.KO = Au moins un champ RuleDuration a une valeur incorrecte. La valeur doit être un entier positif ou nul, ou être indiquée unlimited)

- L'association de RuleDuration et de RuleMeasurement n'est pas inférieure ou égale à 999 ans (STP_IMPORT_RULES_WRONG_TOTALDURATION.KO = L'association de RuleDuration et de RuleMeasurement doit être inférieure ou égale à 999 ans)
- Des règles supprimées sont actuellement utilisées (STP_IMPORT_RULES_DELETE_USED_RULES.KO = Des régles supprimées sont actuellement utilisées)
- Des durées sont inférieures ou égales aux durées minimales autorisées dans la configuration de la plateforme (STP_IMPORT_RULES_RULEDURATION_EXCEED.KO = Échec lors du contrôle de sécurité des règles de gestion. Les durées des règles de gestion doivent être supérieures ou égales aux durées minimales requises par le tenant)
- FATAL : une erreur technique est survenue lors du contrôle des règles de gestion (CHECK_RULES.FATAL=Erreur technique lors du contrôle de la conformité du fichier de règles de gestion)

5.2.1.2. Génération du rapport d'analyse du référentiel des règles de gestion RULES_REPORT (RulesManagerFileImpl.java)

- **Règle** : tâche consistant à créer le rapport d'import des règles
- **Type** : bloquant
- **Statuts** :
  - OK : le rapport est généré (RULES_REPORT.OK = Succès de la génération du rapport d'analyse du référentiel des règles de gestion)
  - KO : pas de cas KO
  - FATAL : une erreur technique est survenue lors de la création du rapport (RULES_REPORT.FATAL = Erreur technique lors de la génération du rapport d'analyse du référentiel des règles de gestion)

5.2.1.3. Persistance des données en base COMMIT_RULES (RulesManagerFileImpl.java)

- **Règle** : tâche consistant à enregistrer les données du référentiel en base
- **Type** : bloquant
- **Statuts** :
  - OK : les données sont persistées en base (COMMIT_RULES.OK = Succès de la persistance des données en base)
  - FATAL : une erreur technique est survenue lors de la persistance des données en base (COMMIT_RULES.FATAL = Erreur technique lors de la persistance des données en base)

5.2.1.4. Enregistrement du fichier d'import du référentiel des règles de gestion STP_IMPORT_RULES_BACKUP_CSV (RulesManagerFileImpl.java)

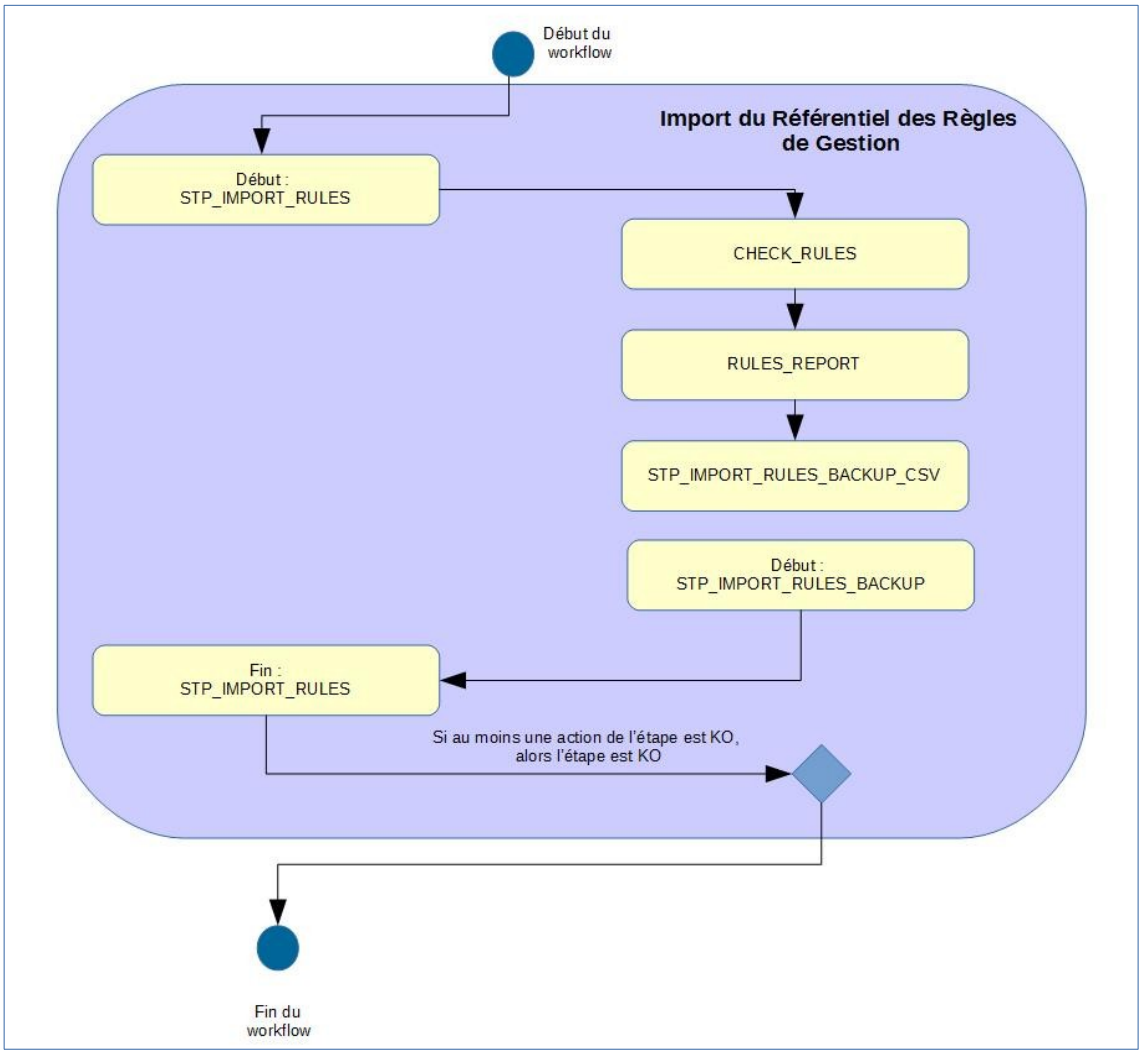
- **Règle** : tâche consistant à écrire le fichier .csv d'import du référentiel des règles de gestion sur les offres de stockage
- **Type** : bloquant
- **Statuts** :
  - OK : le CSV d'import est enregistré (STP_IMPORT_RULES_BACKUP_CSV.OK = Succès du processus d'enregistrement du fichier d'import du référentiel des règles de gestion)
  - KO : pas de cas KO
  - FATAL : une erreur technique est survenue lors de l'enregistrement du CSV d'import (STP_IMPORT_RULES_BACKUP_CSV.FATAL = Erreur technique lors du processus d'enregistrement du fichier d'import du référentiel des règles de gestion)

5.2.1.5. Sauvegarde du JSON STP_IMPORT_RULES_BACKUP (RulesManagerFileImpl.java)

- **Règle** : tâche consistant à écrire le fichier .json correspondant à une copie de la base de données sur les offres de stockage
- **Type** : bloquant
- **Statuts** :
  - OK : une copie de la base de donnée nouvellement importée est enregistrée (STP_IMPORT_RULES_BACKUP.OK = Succès du processus de sauvegarde du référentiel des règles de gestion)
  - KO : pas de cas KO
  - FATAL : une erreur technique est survenue lors de la copie de la base de donnée nouvellement importée (STP_IMPORT_RULES_BACKUP.FATAL = Erreur technique lors du processus de sauvegarde du référentiel des règles de gestion)

5.2.2. Structure du Workflow d'import d'un référentiel des règles de gestion

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



5.2.3.Structure du rapport d'entrée du référentiel des règles de gestion

Lorsqu'un nouveau référentiel est importé, la solution logicielle Vitam génère un rapport de l'opération. Ce rapport est en 2 parties :

- « Operation » contient :
  - « evId » : l'identifiant de l'opération.
  - « evDateTime » : la date et l'heure de l'opération d'import.
  - « evType » : le type d'opération. Dans le cadre de ce rapport, il s'agit toujours de « STP_IMPORT_RULES ».
  - « outMessg » : message final de l'opération (Succès/Avertissement/Échec du processus d'import du référentiel des règles de gestion)
- Le détail des opérations effectuées :
  - « updatedRules » : contient l'intégralité des règles dont une mise à jour a été effectuée.
  - « deletedRules » : contient l'intégralité des règles dont une mise à jour a été effectuée.
  - « usedFileRulesToUpdate » : contient l'intégralité des règles en cours d'utilisation dont une mise à jour a été effectuée. Chaque détail précise en plus l'ensemble des informations relatives aux règles de gestion mises à jour, dans la version mise à jour de la règle (identifiant, type, intitulé, description, durée et mesure).
  - « usedFileRulesToDelete » : contient l'intégralité des règles en cours d'utilisation dont la suppression a été demandée lors de la mise à jour du référentiel des règles de gestion. Chaque détail précise en plus l'ensemble des informations relatives aux règles de gestion mises à jour, dans la version mise à jour de la règle (identifiant, type, intitulé, description, durée et mesure).
  - « error » : liste des erreurs rencontrées lors d'un import ou d'une mise à jour, ligne par ligne. Pour chaque ligne du fichier en erreur, le rapport détaille les éléments suivants :
    - « Code » : type de rapport dans la base de données MongoDB
    - « Message » : message exact de l'erreur ou du résultat d'exécution
    - « Information additionnelle » : précision supplémentaire concernant l'erreur rencontrée.

5.2.3.1. Exemples

Exemple 1 : import initial d'un référentiel en succès

Le rapport généré est :


```
"FileRulesToImport": ["APP-00001", "APP-00002", "APP-09002", "APP-09048", "APP-09049",
"APP-09050", "APP-09051", "ACC-00001", "ACC-00002", "ACC-00003", "ACC-00004", "ACC-00005",
"ACC-00006", "ACC-00007", "ACC-00008", "ACC-00009", "ACC-00010", "ACC-00011", "ACC-00012",
"ACC-00013", "ACC-00014", "ACC-00015", "ACC-00016", "ACC-00017", "ACC-00018", "ACC-00019",
"ACC-00020", "ACC-00021", "ACC-00022", "ACC-00023", "ACC-00024", "ACC-00025", "ACC-00026",
"ACC-00027", "ACC-00028", "ACC-00029", "ACC-00030", "ACC-00031", "ACC-00032", "ACC-00033",
"ACC-00034", "ACC-00035", "ACC-00036", "STO-00001", "STO-00002", "STO-00003", "R1", "R2",
"R3", "R4", "R5", "R6", "DIS-00001", "DIS-00002", "DIS-00003", "DIS-00004", "REU-00001",
"CLASS-00001", "APP-1069001", "APP-1069002", "APP-1069003", "APP-1069004", "DIS-1069001"],

  "Operation": {

     "evId": "aeeaaaaabchia7juabddealol3qdydiaaaaq",

     "evDateTime": "2019-11-12T09:09:50.676",

     "evType": "STP_IMPORT_RULES",

     "outMessg": "Succès du processus d'enregistrement de la copie du référentiel des
règles de gestion"
```


Exemple 2 : mise à jour d'un référentiel existant en échec

Dans cet exemple, la mise à jour :

- Essaye de modifier une RuleType d'une règle en lui mettant « AccessRulez » au lieu de « AccessRule »
- Met à jour une règle de gestion en cours d'utilisation

Le rapport généré est :


```
{
      "Operation": {

     "evId": "aeeaaaaabchia7juabddealol42lhiaaaaaq",

     "evDateTime": "2019-11-12T10:42:01.394",

     "evType": "STP_IMPORT_RULES",

     "outMessg": "Échec du processus d'import du référentiel des règles de gestion"
      },
      "error": {
              "line 6": [{
                      "Code": "STP_IMPORT_RULES_WRONG_RULETYPE_UNKNOW.KO",
                      "Message": "Au moins une RuleType est incorrecte. RuleType
autorisés : AppraisalRule, AccessRule, StorageRule, DisseminationRule, ReuseRule,
ClassificationRule",
                      "Information additionnelle": "AccessRulez"
              }]
      }

}
```

5.3.Workflow d'administration d'un référentiel des formats

Cette section décrit le processus (workflow) permettant d'administrer un référentiel des formats. Cette opération n'est réalisable que sur le tenant d'administration.

Toutes les étapes, tâches et traitements sont journalisés dans le journal des opérations et décrivent le processus (clé et description de la clé associée dans le journal des opérations) tel qu'implémenté dans la version actuelle de la solution logicielle Vitam.

5.3.1.Processus d'import et de mise à jour d'un référentiel de formats (STP_REFERENTIAL_FORMAT_IMPORT)

L'import du référentiel des formats permet de vérifier le formalise de ce dernier, notamment que le fichier doit être au format xml et respecter le formalisme du référentiel PRONOM publié par the National Archives (UK) c'est-à-dire que chaque format contient bien le nombre d'informations minimales attendues. Tous les éléments réalisés au cours de ce processus sont exécutés dans une seule étape.

Ce processus d'import débute lors du lancement du téléchargement dans la solution logicielle Vitam du fichier XML contenant le référentiel.

La fin du processus peut prendre plusieurs statuts

- **Statuts** :
  - OK : les informations correspondant à chaque format sont décrites conformément aux règles émises pour décrire le référentiel PRONOM (STP_REFERENTIAL_FORMAT_IMPORT.OK = Succès du processus d'import du référentiel des formats).
  - KO : la condition ci-dessus n'est pas respectée (STP_REFERENTIAL_FORMAT_IMPORT.KO = Échec du processus d'import du référentiel des formats).
  - WARNING : la version et/ou la date du référentiel PRONOM est/sont validée(s) mais un avertissement signale que la version et/ou la date du référentiel est/sont antérieure(s) à celle(s) du référentiel présent dans la solution logicielle Vitam ou que celle(s)-ci est/sont identique(s) (STP_REFERENTIAL_FORMAT_IMPORT.WARNING = Avertissement lors du processus d'import du référentiel des formats version (n°) du fichier de signature PRONOM)
  - FATAL : une erreur technique est survenue lors de l'import du référentiel des formats (STP_REFERENTIAL_FORMAT_IMPORT.FATAL = Erreur technique lors du processus d'import du référentiel des formats)

5.3.1.1. Processus de sauvegarde du référentiel des formats STP_BACKUP_REFERENTIAL_FORMAT

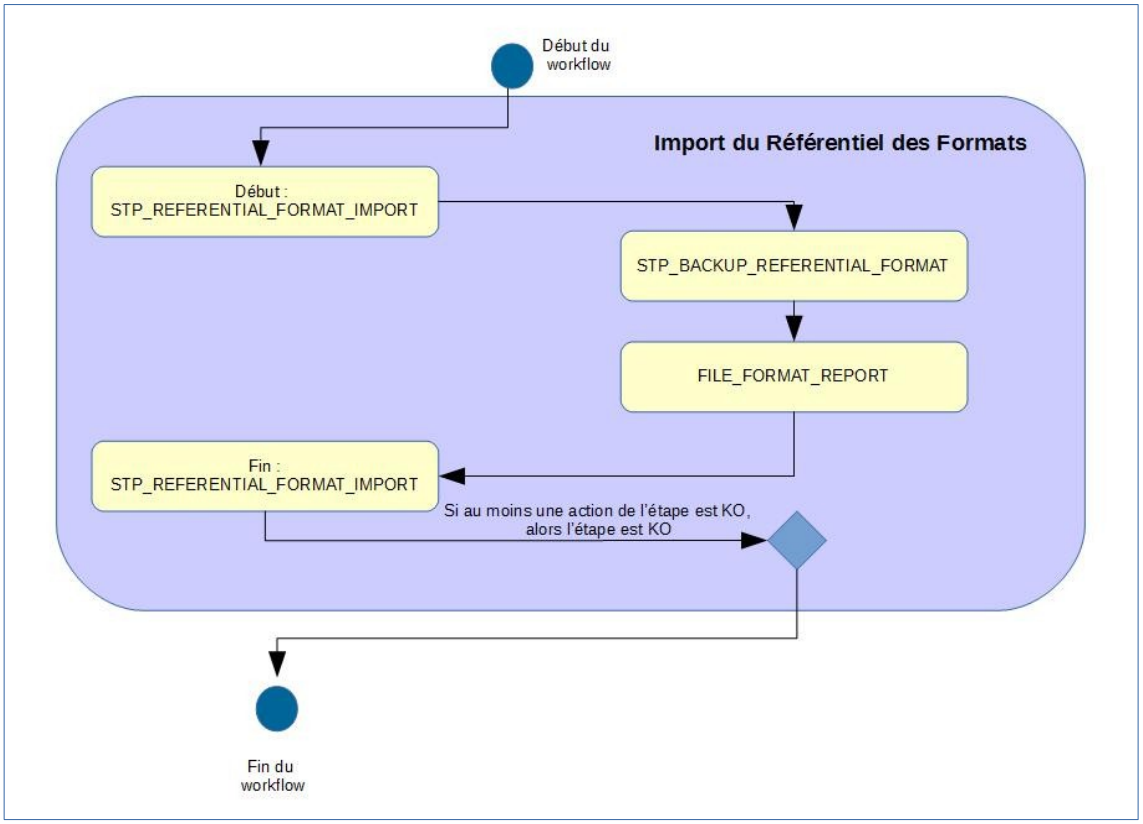
- **Règle** : tâche consistant à sauvegarder le référentiel des formats sur les offres de stockage.
- **Type** : bloquant
- **Statuts** :
  - OK : la sauvegarde du référentiel des formats a bien été effectuée (STP_BACKUP_REFERENTIAL_FORMAT.OK = Succès du processus de sauvegarde du référentiel des formats)
  - KO : la sauvegarde du référentiel des formats n'a pas été effectuée (STP_BACKUP_REFERENTIAL_FORMAT.KO = Échec du processus de sauvegarde du référentiel des formats)
  - FATAL : une erreur technique est survenue lors de la sauvegarde du référentiel des formats (STP_BACKUP_REFERENTIAL_FORMAT.FATAL = Erreur technique lors du processus de sauvegarde du référentiel des formats)

5.3.1.2. Processus de génération du rapport d'import du référentiel des formats FILE_FORMAT_REPORT

- **Règle** : tâche consistant à créer le rapport d'import du référentiel des formats.
- **Type** : bloquant
- **Statuts** :
  - OK : le rapport d'import du référentiel des formats a bien été créé (FILE_FORMAT_REPORT.OK = Succès du processus de génération du rapport d'import du référentiel des formats)
  - KO : pas de KO
  - FATAL : une erreur technique est survenue lors de la création du rapport d'import référentiel des formats (FILE_FORMAT_REPORT.FATAL = Erreur technique lors du processus de génération du rapport du référentiel des formats)

5.3.2. Structure du Workflow d'import d'un référentiel des formats

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



5.3.3. Structure du rapport d'administration d'un référentiel des formats

Lorsqu'un nouveau référentiel est importé, la solution logicielle Vitam génère un rapport de l'opération. Ce rapport est en plusieurs parties :

- « Operation » contient :
  - evType : le type d'opération. Dans le cadre de ce rapport, il s'agit toujours de « STP_REFERENTIAL_FORMAT_IMPORT »
  - evDateTime : la date et l'heure du début de la génération du rapport
  - evId : l'identifiant de l'opération

- « StatusCode » : le statut de l'opération (OK, KO, WARNING)
- « PreviousPronomVersion » : le numéro de la version du référentiel précédemment installé dans la solution logicielle Vitam
- « PreviousPronomCreationDate » : la date de créationdu référentiel précédemment installé dans la solution logicielle Vitam
- « NewPronomVersion » : le numéro de version du référentiel désormais installé
- « NewPronomCreationDate » : la date de création du référentieldésormais installé
- « RemovedPUIDs » : la liste des PUID qui ont été supprimés
- « AddedPUID »s : la liste des PUID qui ont été ajoutés
- « UpdatedPUIDs » : la liste des PUID qui ont été mis à jour, accompagnés d'un différentiel entre les métadonnées précédemment importées et les nouvelles
- « Warnings » : le message concernant le warning : le référentiel des formats importé est plus ancien, en termes de numéro de version et/ou de date de création, que la version du référentiel présente dans la solution, le référentiel des formats importé est identique, en termes de numéro de version et/ou de date de création, à la version précédemment installée dans la solution.

Exemple :


```
« Operation » ({}

« evType » : « STP_REFERENTIAL_FORMAT_IMPORT », « evDateTime » : « 2018-12-
11T14:46:43.856 », « evId » : « aeeaaaaaaghg7kglaboimalhtw57z7qaaaaq »

}, « StatusCode » : « WARNING », « PreviousPronomVersion » : « 94 »,
« PreviousPronomCreationDate » : « 2018-09-17T12:54:53.000 », « NewPronomVersion » :
« 88 », « NewPronomCreationDate » : « 2016-09-27T15:37:53.000 », « RemovedPUIDs » :
[ « fmt/1216 », « fmt/1217 », « fmt/1214 », « fmt/1215 », « fmt/1212 », « fmt/1213 »,
« fmt/1210 », « fmt/1211 », « fmt/1108 », « fmt/1109 », « fmt/1106 », « fmt/1107 »,
« fmt/1104 », « fmt/1105 », « fmt/1102 », « fmt/1103 », « fmt/1100 », « fmt/1101 »,
« fmt/985 », « fmt/986 », « fmt/987 », « fmt/988 », « fmt/981 », « fmt/982 »,
« fmt/983 », « fmt/984 », « fmt/989 », « fmt/980 », « fmt/975 », « fmt/976 »,
« fmt/977 », « fmt/978 », « fmt/979 », « fmt/1209 », « fmt/1207 », « fmt/1208 »,
« fmt/1205 », « fmt/1206 », « fmt/1203 », « fmt/1204 », « fmt/1201 », « fmt/1202 »,
« fmt/1200 », « fmt/1140 », « fmt/1020 », « fmt/1141 », « fmt/1018 », « fmt/1139 »,
« fmt/1019 », « fmt/1016 », « fmt/1137 », « fmt/1017 », « fmt/1138 », « fmt/1014 »,
« fmt/1135 », « fmt/1015 », « fmt/1136 », « fmt/1012 », « fmt/1133 », « fmt/1013 »,
« fmt/1134 », « fmt/1010 », « fmt/1131 », « fmt/1011 », « fmt/1132 », « fmt/1030 »,
« fmt/1151 », « fmt/996 », « fmt/1031 », « fmt/1152 », « fmt/997 », « fmt/998 »,
« fmt/1150 », « fmt/999 », « fmt/992 », « fmt/993 », « fmt/994 », « fmt/995 »,
« fmt/1029 », « fmt/1027 », « fmt/1148 », « fmt/1028 », « fmt/1149 », « fmt/1025 »,
« fmt/1146 », « fmt/990 », « fmt/1026 », « fmt/1147 », « fmt/991 », « fmt/1023 », « fmt/
1144 », « fmt/1024 », « fmt/1145 », « fmt/1021 », « fmt/1142 », « fmt/1022 »,
« fmt/1143 », « fmt/1119 », « fmt/1117 », « fmt/1118 », « fmt/1115 », « fmt/1116 »,
« fmt/1113 », « fmt/1114 », « fmt/1111 », « fmt/1112 », « fmt/1110 », « fmt/1130 »,
« fmt/1009 », « fmt/1007 », « fmt/1128 », « fmt/1008 », « fmt/1129 », « fmt/1005 »,
« fmt/1126 », « fmt/1006 », « fmt/1127 », « fmt/1003 », « fmt/1124 », « fmt/1004 »,
« fmt/1125 », « fmt/1001 », « fmt/1122 », « fmt/1002 », « fmt/1123 », « fmt/1120 »,
« fmt/1000 », « fmt/1121 », « fmt/1063 », « fmt/1184 », « fmt/1064 », « fmt/1185 »,
« fmt/1061 », « fmt/206 », « fmt/1182 », « fmt/1062 », « fmt/1183 », « fmt/1180 »,
« fmt/1060 », « fmt/1181 », « fmt/1058 », « fmt/1179 », « fmt/1059 », « fmt/1056 »,
« fmt/1177 », « fmt/1057 », « fmt/1178 », « fmt/1054 », « fmt/1175 », « fmt/1055 »,
« fmt/1176 », « fmt/1074 », « fmt/1195 », « fmt/1075 », « fmt/1196 », « fmt/1072 »,
« fmt/1193 », « fmt/1073 », « fmt/1194 », « fmt/1070 », « fmt/1191 », « fmt/1071 »,
« fmt/1192 », « fmt/1190 », « fmt/1069 », « fmt/1067 », « fmt/1188 », « fmt/1068 »,
« fmt/1189 », « fmt/1065 », « fmt/1186 », « fmt/1066 », « fmt/1187 », « fmt/1041 »,
« fmt/1162 », « fmt/1042 », « fmt/1163 », « fmt/1160 », « fmt/1040 », « fmt/1161 »,
« fmt/1038 », « fmt/1159 », « fmt/1039 », « fmt/1036 », « fmt/1157 », « fmt/1037 »,
« fmt/1158 », « fmt/1034 », « fmt/1155 », « fmt/1035 », « fmt/1156 », « fmt/1032 »,
« fmt/1153 », « fmt/1033 », « fmt/1154 », « fmt/1052 », « fmt/1173 », « fmt/1053 »,
« fmt/1174 », « fmt/1050 », « fmt/1171 », « fmt/1051 », « fmt/1172 », « fmt/1170 »,
« fmt/1049 », « fmt/1047 », « fmt/1168 », « fmt/1048 », « fmt/1169 », « fmt/1045 »,
« fmt/1166 », « fmt/1046 », « fmt/1167 », « fmt/1043 », « fmt/1164 », « fmt/1044 »,
```

```
« fmt/1165 », « fmt/1098 », « fmt/1099 », « fmt/1085 », « fmt/1086 », « fmt/1083 »,
« fmt/1084 », « fmt/1081 », « fmt/1082 », « fmt/1080 », « fmt/1078 », « fmt/1199 »,
« fmt/1079 », « fmt/1076 », « fmt/1197 », « fmt/1077 », « fmt/1198 », « fmt/1096 »,
« fmt/1097 », « fmt/1094 », « fmt/1095 », « fmt/1092 », « fmt/1093 », « fmt/1090 »,
« fmt/1091 », « fmt/1089 », « fmt/1087 », « fmt/1088 » ], « AddedPUIDs » : [ ],
« UpdatedPUIDs » : {

« fmt/563 » : [ « + MimeType : « , « - MimeType : application/postscript » ],
« fmt/641 » : [ « + HasPriorityOverFileFormatID : [ fmt/154 ] », « -
HasPriorityOverFileFormatID : [ fmt/154, fmt/353 ] » ], « fmt/245 » : [ « + Extension :
[ \n ] », « - Extension : [ ] » ], « fmt/899 » : [ « + MimeType : application/octet-
stream », « - MimeType : application/vnd.microsoft.portable-executable » ],
« x-fmt/430 » : [ « + Extension : [ msg ] », « - Extension : [ msg, oft ] » ],
« fmt/417 » : [ « + MimeType : « , « - MimeType : application/postscript » ],
« fmt/616 » : [ « + MimeType : application/font-woff », « + Version : « , « - MimeType :
font/woff », « - Version : 1.0 » ], « fmt/418 » : [ « + MimeType : « , « - MimeType :
application/postscript » ], « fmt/419 » : [ « + MimeType : « , « - MimeType :
application/postscript » ], « fmt/570 » : [ « + HasPriorityOverFileFormatID : [ ] », « -
HasPriorityOverFileFormatID : [ fmt/986 ] » ]

}, « Warnings » : [ "Same referential version: 94", "Same referential date: 2018-09-
17T12:54:53.000", "1669 puids removed." ]

}
```

5.4.Workflow d'administration d'un référentiel des services agents

Cette section décrit le processus (workflow) permettant d'importer un référentiel de services agents.

Toutes les étapes, tâches et traitements sont journalisées dans le journal des opérations. Et décrivent le processus (clé et description de la clé associée dans le journal des opérations) tel qu'implémenté dans la version actuelle de la solution logicielle Vitam.

L'import d'un référentiel de services agents permet de vérifier le formalisme de ce dernier, notamment que les données obligatoires sont bien présentes pour chacun des agents. Tous les éléments réalisés au cours de ce processus sont exécutés dans une seule étape. Cet import concerne aussi bien l'import initial (pas de services agents pré-existant) que la mise à jour du référentiel.

Ce processus d'import débute lors du lancement du téléchargement dans la solution logicielle Vitam du fichier CSV contenant le référentiel.

La fin du processus peut prendre plusieurs statuts :

5.4.1.Processus d'import et mise à jour d'un référentiel de services agents (STP_IMPORT_AGENCIES)

- **Règle :** étape consistant à vérifier que le fichier importé remplit les conditions suivantes :
  - il est au format CSV
  - les informations suivantes sont toutes décrites dans l'ordre exact pour chacun des services agents :
    - Identifier
    - Name
    - Description (optionnel)
    - L'identifiant doit être unique
- **Type :** bloquant
- **Statuts** :
  - OK : le fichier respecte les règles (STP_IMPORT_AGENCIES.OK = Succès du processus d'import du référentiel des services agents)
  - KO :
    - Cas 1 : une information concernant les services agents est manquante (Identifier, Name, Description) (STP_IMPORT_AGENCIES.KO = Échec du processus d'import du référentiel des services agents). .
    - Cas 2 : un service agent qui était présent dans la base a été supprimé (STP_IMPORT_AGENCIES.DELETION.KO = Échec du processus d'import du référentiel des services agents : Des services agents absents du fichier d'import sont utilisés par au moins une unité archivistique présente dans le système)
  - FATAL : une erreur technique est survenue lors de l'import du référentiel des services agents (STP_IMPORT_AGENCIES.FATAL = Erreur technique lors du processus d'import du référentiel des services agents)

5.4.1.1. Vérification des contrats utilisés IMPORT_AGENCIES.USED_CONTRACT

- **Règle** : tâche consistant à contrôler les contrats d'accès utilisant des services agents modifiés
- **Type** : bloquant
- **Statuts** :
  - OK : aucun des services agents utilisés par des contrats d'accès n'a été modifié (STP_IMPORT_AGENCIES.USED_CONTRACT.OK = Succès du processus de vérification des services agents utilisés dans les contrats d'accès)

- WARNING : un ou plusieurs services agents utilisés par des contrats d'accès ont été modifiés (STP_IMPORT_AGENCIES.USED_CONTRACT.WARNING = Avertissement lors du processus de vérification des services agents utilisés dans les contrats d'accès)
- KO : pas de cas KO
- FATAL : une erreur technique est survenue lors de la vérification des services agents utilisés dans les contrats d'accès (STP_IMPORT_AGENCIES.USED_CONTRACT.FATAL = Erreur technique lors du processus de vérification des services agents utilisés dans les contrats d'accès)

5.4.1.2. Vérification des unités archivistiques IMPORT_AGENCIES.USED_AU

- **Règle** : tâche consistant à contrôler les unités archivistiques référençant des services agents modifiés
- **Type** : bloquant
- **Statuts** :
  - OK : aucun service agent référencé par les unités archivistiques n'a été modifié (STP_IMPORT_AGENCIES.USED_AU.OK = Succès du processus de vérification des services agents référencés par les unités archivistiques)
  - WARNING : au moins un service agent référencé par une unité archivistique a été modifié (STP_IMPORT_AGENCIES.USED_AU.WARNING = Avertissement lors du processus de vérification des services agents référencés par les unités archivistiques)
  - KO : pas de cas KO
  - FATAL : une erreur technique est survenue lors de la vérification des services agents utilisés par les unités archivistiques (STP_IMPORT_AGENCIES.USED_AU.FATAL = Erreur technique lors du processus de vérification des services agents référencés par les unités archivistiques)

5.4.1.3. Création du rapport au format JSON AGENCIES_REPORT (AgenciesService.java)

- **Règle** : tâche consistant à créer le rapport d'import de référentiel des services agents
- **Type** : bloquant
- **Statuts** :
  - OK : le rapport d'import du référentiel des services agents a bien été créé (STP_AGENCIES_REPORT.OK = Succès du processus de génération du rapport d'import du référentiel des services agents)
  - KO : pas de cas KO
  - FATAL : une erreur technique est survenue lors de la création du rapport d'import du référentiel des services agents (STP_AGENCIES_REPORT.FATAL = Erreur technique lors du processus de génération du rapport d'import du référentiel des services agents)

5.4.1.4. Sauvegarde du CSV d'import IMPORT_AGENCIES_BACKUP_CSV (AgenciesService.java)

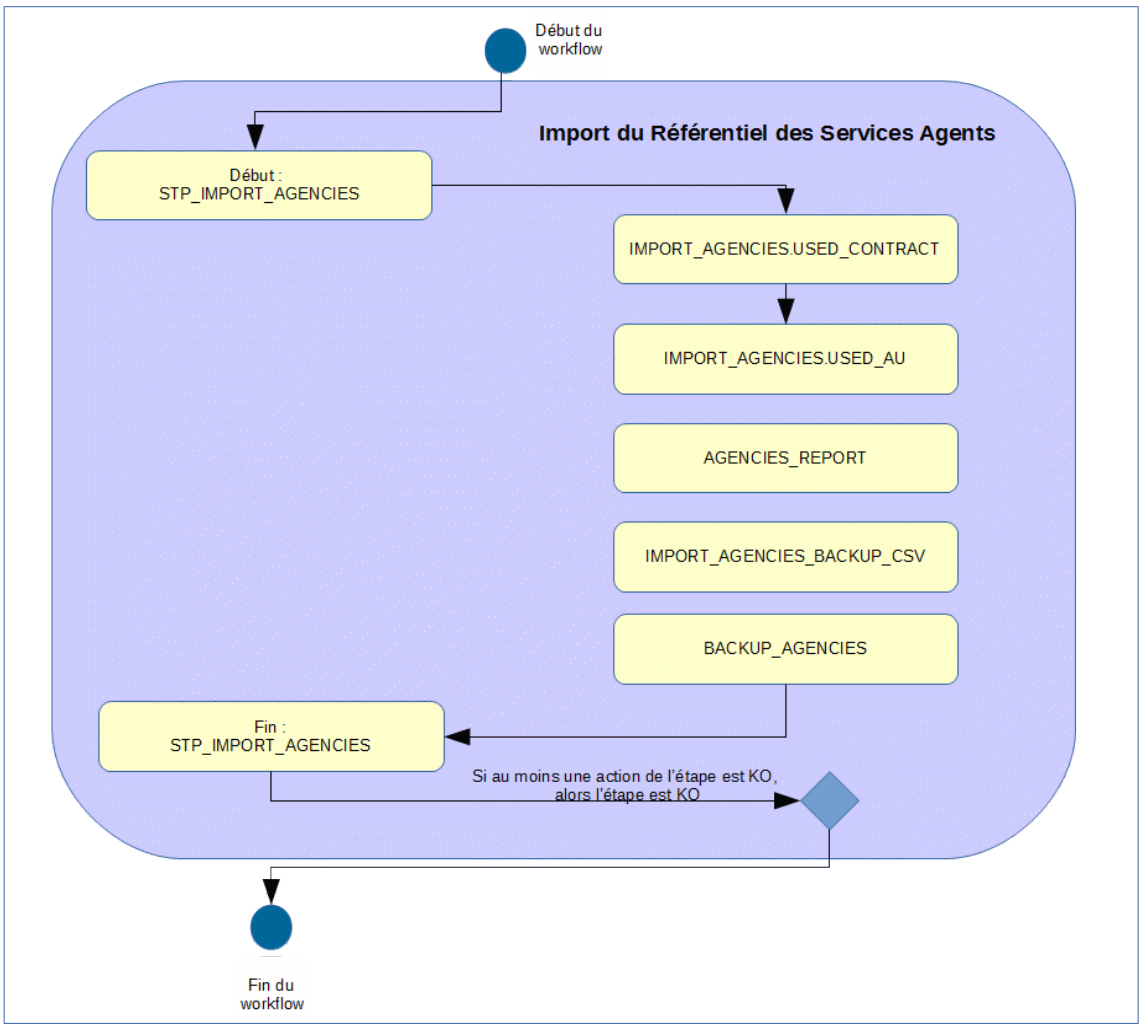
- **Règle** : tâche consistant à sauvegarder le fichier d'import de référentiel des services agents sur les offres de stockage
- **Type** : bloquant
- **Statuts** :
  - OK : le fichier d'import du référentiel des services agent a bien été sauvegardé (STP_IMPORT_AGENCIES_BACKUP_CSV.OK = Succès du processus de sauvegarde du fichier d'import du référentiel des services agents)
  - KO : pas de cas KO
  - FATAL : une erreur technique est survenue lors de la sauvegarde de fichier d'import du référentiel des services agents (STP_AGENCIES_REPORT.FATAL = Erreur technique lors du processus de sauvegarde du fichier d'import du référentiel des services agents)

5.4.1.5. Sauvegarde d'une copie de la base de données BACKUP_AGENCIES (AgenciesService.java)

- **Règle** : tâche consistant à créer une copie de la base de données contenant le référentiel des services agents sur les offres de stockage
- **Type** : bloquant
- **Statuts** :
  - OK : la copie de la base de donnée contenant le référentiel des services agents a été créé avec succès (STP_BACKUP_AGENCIES.OK = Succès du processus de sauvegarde du référentiel des services agents)
  - KO : pas de cas KO
  - FATAL : une erreur technique est survenue lors de la création d'une copie de la base de données contenant le référentiel des services agent (STP_BACKUP_AGENCIES.FATAL = Erreur technique lors du processus de sauvegarde du référentiel des services agents)

5.4.2. Structure du Workflow d'import d'un référentiel des services agents

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



5.4.3.Structure du rapport d'import du référentiel des services agents

Lorsqu'un nouveau référentiel est importé, la solution logicielle Vitam génère un rapport de l'opération. Ce rapport est en plusieurs parties :

- « Operation » contient :
  - evType : le type d'opération. Dans le cadre de ce rapport, il s'agit toujours de « STP_IMPORT_AGENCIES »
  - evDateTime : la date et l'heure de l'opération d'import
  - evId : l'identifiant de l'opération
- « AgenciesToImport » : contient la liste des identifiants contenue dans le fichier
- « InsertAgencies » : contient les identifiants des services agents ajoutés
- « UpdatedAgencies » : liste les identifiants des services agents modifiés
- « UsedAgencies By Contrat » : liste les identifiants des services agents modifiés qui sont ou pas utilisés par des contrats d'accès
- « UsedAgencies By AU » : liste les identifiants des services agents modifiés qui sont ou pas ont utilisés dans des unités archivistiques
- « UsedAgencies to Delete » : liste les identifiants des services agents supprimés qui sont pas non utilisés dans des unités archivistiques

- **Exemple 1 : modification et suppression d'un service agent**

Le rapport généré est :


```
{

  "AgenciesToImport": ["test", "Identifier0", "Identifier1", "Identifier2", "Identifier3",
"Identifier4", "Identifier5", "Identifier6", "Identifier7", "Identifier8", "Identifier9",
"Identifier10", "Vitam", "FRAN_NP_009913", "FRAN_NP_009941", "producteur1",
"FRAN_NP_050500", "FRAN_NP_050770", "BBBBBB", "FRAN_NP_050758", "FRAN_NP_050056",
"FRAN_NP_005568", "ABCDEFG", "OAIAJJAZ", "ZAJFKNNDSC", "Service_producteur",
"SOC_ARCHEO_TOURRAINE", "EEEEEE", "FRAN_NP_050392", "FRAN_NP_050313", "FRAN_NP_051314",
"FR_ORG_AGEN", "FRAAA", "CCCCC", "ASP", "producteur2", "FRAN_NP_000001", "FRAN_NP_005061",
"FRAN_NP_009915", "KARINE_GILBERT", "MICHEL_MERCIER", "DGFIP", "FRAN_NP_005164",
"SOC_ARCHEO_TOURAINE", "Service_versant", "XXXXXXX", "YYYYYYY", "FRAN_NP_009196",
"FRAN_NP_005761", "FFFFFF", "FRAN_NP_005479", "FRAN_NP_000992", "FRAN_NP_000002",
"FRAN_NP_000013", "FRAN_NP_000015", "FRAN_NP_000008", "FRAN_NP_000017", "FRAN_NP_000024",
"FRAN_NP_000018", "FRAN_NP_000023", "FRAN_NP_000010", "FRAN_NP_000019", "FRAN_NP_000020",
"FRAN_NP_000021", "FRAN_NP_000022", "FRAN_NP_000009", "RATP"],

  "UsedAgencies to Delete": ["FRAN_NP_009734", "Test_VITAM"],

  "Operation": {

     "evId": "aeeaaaaabchia7juabddealol5nlueaaaaaq",

     "evDateTime": "2019-11-12T11:23:36.061",

     "evType": "IMPORT_AGENCIES"

  }

}
```


Exemple 2 : ajout en erreur d'un service agent, causé par un champ obligatoire qui est manquant

Le rapport généré est :


```
{
      "Operation": {
    "evId":"aecaaaaaacflvhgbabrs6alb6vdoehyaaaaq",
              "evType": "STP_IMPORT_AGENCIES",
              "evDateTime": "2017-11-02T15:36:03.976"
```

```
      },
      "AgenciesToImport": ["AG-TNR0002"],
      "UsedAgencies to Delete":["AG-TNR0002"]
}
```

5.5.Workflow d'administration d'un référentiel des contrats d'entrée

Cette section décrit le processus (workflow) permettant d'importer et de mettre à jour un contrat d'entrée.

Toutes les étapes, tâches et traitements sont journalisées dans le journal des opérations et décrivent le processus (clé et description de la clé associée dans le journal des opérations) tel qu'implémenté dans la version actuelle de la solution logicielle Vitam.

5.5.1.Processus d'import d'un contrat d'entrée (STP_IMPORT_INGEST_CONTRACT)

Le processus d'import d'un contrat d'entrée permet à la fois de vérifier qu'il contient les informations minimales obligatoires, que l'ensemble des informations sont cohérentes, et de lui affecter des éléments peuplés automatiquement.

Tous les éléments réalisés au cours de ce processus sont exécutés dans une seule étape.

- **Règle** : étape consistant à vérifier la présence des informations minimales dans le contrat d'entrée, à s'assurer de la cohérence des informations saisies, à affecter des données aux champs peuplés par la solution logicielle Vitam et à enregistrer le contrat :
  - Le champ « Name » est peuplé d'une chaîne de caractères
  - Si le tenant concerné est en mode « esclave », le champ « Identifier » doit être rempli. Sinon, il est automatiquement complété par la solution logicielle Vitam
  - Les données suivantes optionnelles si elles sont remplies le sont en respectant les règles énoncées pour chacune :
    - Le champ « Description » doit être une chaîne de caractères
    - Le champ « Status » doit avoir la valeur ACTIVE ou INACTIVE
    - Le champ « ArchiveProfile » doit être un tableau d'une ou plusieurs chaînes de caractère. Chacune de ces chaînes de caractère doit correspondre au champ « Identifier » d'un profil d'archivage contenu dans le référentiel des profils
    - Le champ « CheckParentLink » : doit avoir la valeur ACTIVE ou INACTIVE
    - Le champ « LinkedParentId » doit être une chaîne de caractères devant correspondre au GUID d'une AU de plan de classement ou d'arbre de positionnement déjà pris en charge par la solution logicielle Vitam sur le même tenant
    - Le champ « MasterMandatory » doit avoir la valeur « true » ou « false »
    - Le champ « EveryDataObjectVersion » doit avoir la valeur « true » ou « false »
    - Le champ « DataObjectVersion » devrait être un tableau dont chaque élément est dans l'énumération suivante (ou être vide) : « BinaryMaster », « TextContent », « Thumbnail », « PhysicalMaster », « Dissemination »
    - Le champ « FormatUnidentifiedAuthorized » doit avoir la valeur « true » ou « false »
    - Le champ « EveryFormatType » doit avoir la valeur « true » ou « false »
    - Le champ « FormatType » doit être un tableau dont chaque élément est une PUID du référentiel des formats (exemple : « fmt/17 »)
    - Le champ « ManagementContractId » est soit vide soit un « Identifier » de contrat de gestion existant dans le référentiel de contrats de gestion
- **Type** : bloquant
- **Statuts** :
  - OK : le contrat répond aux exigences des règles (STP_IMPORT_INGEST_CONTRACT.OK = Succès du processus d'import du contrat d'entrée)
  - KO : une des règles ci-dessus n'a pas été respectée (STP_IMPORT_INGEST_CONTRACT.KO =

Échec du processus d'import du contrat d'entrée)

- Cas n°1 : l'identifiant utilisé existe déjà (STP_IMPORT_INGEST_CONTRACT.IDENTIFIER_DUPLICATION.KO = Échec de l'import : l'identifiant est déjà utilisé)
- Cas n°2 : un champ obligatoire n'est pas renseigné (STP_IMPORT_INGEST_CONTRACT.EMPTY_REQUIRED_FIELD.KO = Échec de l'import : au moins un des champs obligatoires n'est pas renseigné)
- Cas n°3 : le profil d'archivage mentionné n'existe pas (STP_IMPORT_INGEST_CONTRACT.PROFILE_NOT_FOUND.KO = Échec de l'import : profil d'archivage non trouvé)
- Cas n°4 : le contrat d'entrée autorise tous les formats mais déclare des formats dans la liste des formats autorisés (STP_IMPORT_INGEST_CONTRACT.FORMAT_MUST_BE_EMPTY.KO= Échec de l'import : la liste blanche des formats doit être vide lorsque tous les formats sont autorisés) ;
- Cas n°5 : un ou plusieurs formats déclarés ne sont pas référencés dans le référentiel des formats (STP_IMPORT_INGEST_CONTRACT.FORMAT_NOT_FOUND.KO = Échec de l'import : un ou plusieurs formats ne sont pas référencés dans le référentiel des formats) ;
- Cas n°6 : le contrat d'entrée n'autorise pas tous les formats mais ne déclare pas de formats dans la liste des formats autorisés (STP_IMPORT_INGEST_CONTRACT.FORMAT_MUST_NOT_BE_EMPTY.KO = Échec de l'import : la liste blanche des formats ne peut pas être vide lorsque tous les formats ne sont pas autorisés)
- Cas n°7 : le contrat de gestion déclaré n'est pas référencé dans le référentiel des contrats de gestion (STP_IMPORT_INGEST_CONTRACT.MANAGEMENT_CONTRAT_NOT_FOUND.KO = Échec de l'import : le contrat de gestion n'est pas référencé dans le référentiel des contrats de gestion) ;

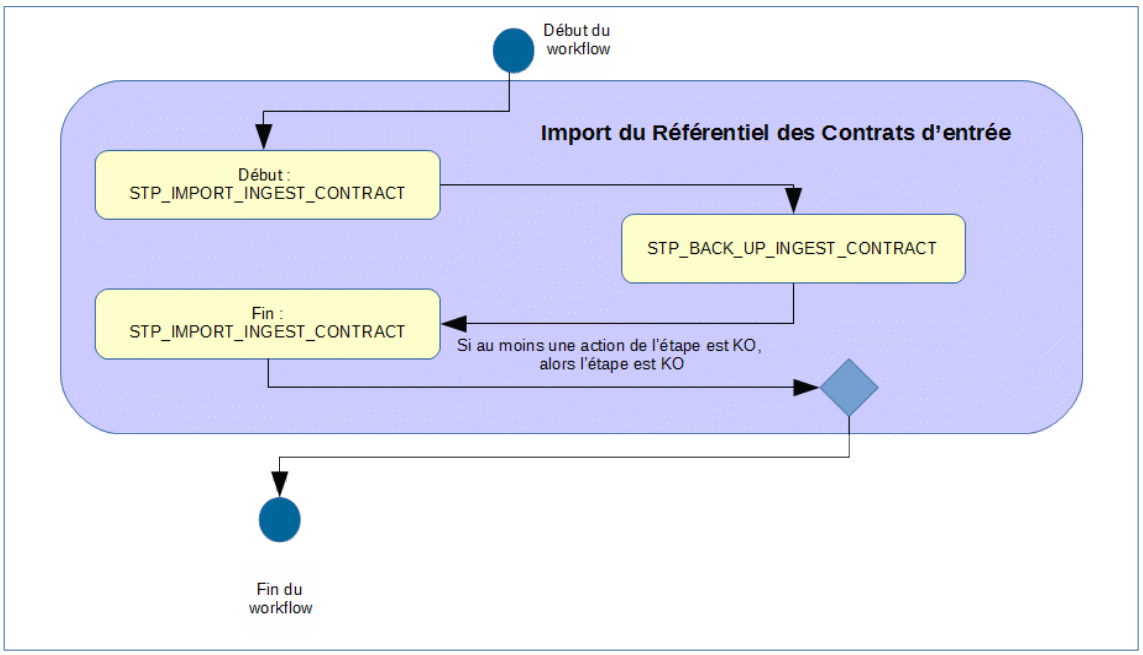
- FATAL : une erreur technique est survenue lors de l'import du contrat (STP_IMPORT_INGEST_CONTRACT.FATAL = Erreur technique du processus d'import du contrat d'entrée)
- WARNING : avertissement lors du processus d'import du contrat d'entrée (STP_IMPORT_INGEST_CONTRACT.WARNING = Avertissement lors du processus d'import du contrat d'entrée)

Sauvegarde du JSON (STP_BACKUP_INGEST_CONTRACT)

- **Règle** : étape consistant à enregistrer une copie de la base de données des contrats d'entrée sur les offres de stockage
- **Type** : bloquant
- **Statuts** :
  - OK : une copie de la base de données nouvellement importée est enregistrée (STP_BACKUP_INGEST_CONTRACT.OK = Succès du processus de sauvegarde des contrats d'entrée)
  - KO : pas de cas KO
  - FATAL : une erreur technique est survenue lors de la copie de la base de données nouvellement importée (STP_BACKUP_INGEST_CONTRACT.FATAL = Erreur technique lors du processus de sauvegarde des contrats d'entrée)

5.5.2. Structure du Workflow d'import d'un référentiel des contrats d'entrée

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



5.5.3. Processus de mise à jour d'un contrat d'entrée (STP_UPDATE_INGEST_CONTRACT)

Le processus de mise d'un contrat d'entrée permet à la fois de vérifier qu'il contient les informations minimales obligatoires, que de vérifier la cohérence de l'ensemble des informations sont cohérentes, et de lui affecter des éléments peuplés automatiquement.

Tous les éléments réalisés au cours de ce processus sont exécutés dans une seule étape.

- **Règle** : opération consistant à modifier un contrat d'entrée, selon un processus identique à celui de l'import initial
- **Type** : bloquant
- **Statuts** :
  - OK : la mise à jour du contrat d'entrée a bien été effectuée (STP_UPDATE_INGEST_CONTRACT.OK = Succès du processus de mise à jour du contrat d'entrée)
  - KO : la mise à jour du contrat d'entrée n'a pas été effectuée (STP_UPDATE_INGEST_CONTRACT.KO = Échec du processus de mise à jour du contrat d'entrée)
    - Cas n°1 : l'identifiant du contrat d'entrée est inconnu (STP_UPDATE_INGEST_CONTRACT.CONTRACT_NOT_FOUND.KO = Échec du processus de mise à jour du contrat d'entrée : contrat d"entrée non trouvé)
    - Cas °2 : l'identifiant du contrat d'entrée mis à jour est déjà utilisé STP_UPDATE_INGEST_CONTRACT.IDENTIFIER_DUPLICATION.KO = Échec du processus de mise à jour du contrat d'entrée : l'identifiant est déjà utilisé
    - Cas n°3 : une des valeurs saisies dans le contrat d'entrée ne correspond pas aux valeurs attendues (STP_UPDATE_INGEST_CONTRACT.NOT_IN_ENUM.KO = Échec du processus de mise à jour du contrat d'entrée : une valeur ne correspond pas aux valeurs attendues)
    - Cas n°4 : le profil d'archivage mentionné n'existe pas (STP_UPDATE_INGEST_CONTRACT.PROFILE_NOT_FOUND.KO = Échec du processus de mise à jour du contrat d'entrée : au moins un profil d'archivage est inconnu)

- Cas n°5 : la requête de mise à jour du contrat d'entrée est mal formatée (STP_UPDATE_INGEST_CONTRACT.BAD_REQUEST.KO = Échec du processus de mise à jour du contrat d'entrée : mauvaise requête)
- Cas n°6 : un ou plusieurs formats déclarés ne sont pas référencés dans le référentiel des formats (STP_UPDATE_INGEST_CONTRACT.FILEFORMAT_NOT_FOUND.KO = Échec du processus de mise à jour du contrat d'entrée : au moins un identifiant de format est inconnu)
- Cas n°7 : le contrat de gestion déclaré n'est pas référencé dans le référentiel des contrats de gestion (STP_UPDATE_INGEST_CONTRACT.MANAGEMENT_CONTRAT_NOT_FOUND.KO = Échec du processus de mise à jour : le contrat de gestion est inconnu)

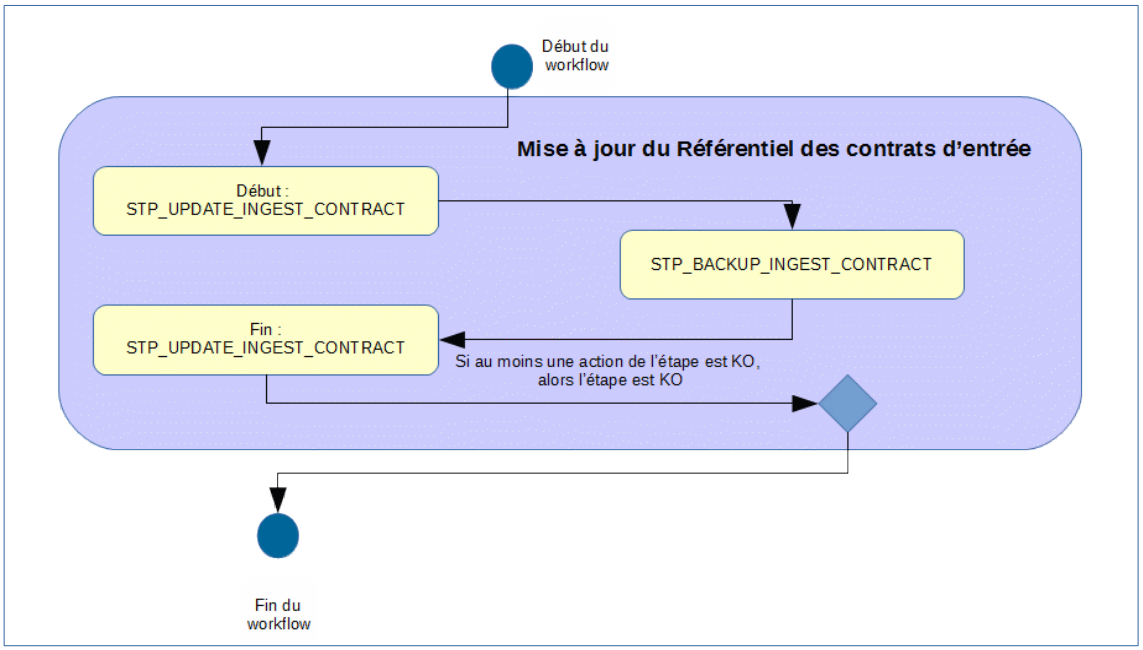
- FATAL : une erreur technique est survenue lors du processus de mise à jour du contrat d'entrée (STP_UPDATE_INGEST_CONTRACT.FATAL = Erreur technique lors du processus de mise à jour du contrat d'entrée)

5.5.3.1. Sauvegarde du JSON (STP_BACKUP_INGEST_CONTRACT)

- **Règle** : tâche consistant à enregistrer une copie de la base de données des contrats d'entrée sur les offres de stockage
- **Type** : bloquant
- **Statuts** :
  - OK : une copie de la base de données nouvellement importée est enregistrée (STP_BACKUP_INGEST_CONTRACT.OK = Succès du processus de sauvegarde des contrats d'entrée)
  - KO : pas de cas KO
  - FATAL : une erreur technique est survenue lors de la copie de la base de données nouvellement importée (STP_BACKUP_INGEST_CONTRACT.FATAL = Erreur technique lors du processus de sauvegarde des contrats d'entrée)

5.5.4.Structure du Workflow de mise à jour d'un référentiel des contrats d'entrée

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



5.6.Workflow d'administration d'un référentiel des contrats d'accès

Cette section décrit le processus (workflow) permettant d'importer et de mettre à jour un contrat d'accès.

Toutes les étapes, tâches et traitements sont journalisés dans le journal des opérations et décrivent le processus (clé et description de la clé associée dans le journal des opérations) tel qu'implémenté dans la version actuelle de la solution logicielle Vitam.

5.6.1.Processus d'import et mise à jour d'un contrat d'accès (STP_IMPORT_ACCESS_CONTRACT)

Le processus d'import d'un contrat d'accès permet à la fois de vérifier qu'il contient les informations minimales obligatoires, de vérifier la cohérence de l'ensemble des informations et de lui affecter des éléments peuplés automatiquement.

Tous les éléments réalisés au cours de ce processus sont exécutés dans une seule étape.

- **Règle :** opération consistant à vérifier la présence des informations minimales dans le contrat d'accès, à s'assurer de la cohérence des informations saisies, à affecter des données aux champs peuplés par la solution logicielle Vitam et et à enregistrer le contrat
  - Les données suivantes sont obligatoirement remplies :
    - Le champ « Name » est peuplé d'une chaîne de caractères
    - Le champ « Identifier » est peuplé d'une chaîne de caractères si le référentiel des contrats d'accès est configuré en mode esclave sur le tenant sélectionné
    - Le champ « _tenant » : identifiant du tenant.
    - Le champ « Status » est peuplé avec la valeur « ACTIVE » ou la valeur « INACTIVE »
    - Le champ « AccessLog » peut être renseigné avec la valeur « ACTIVE » ou la valeur « INACTIVE »
  - Les données suivantes optionnelles, si elles sont remplies, le sont en respectant les règles énoncées pour chacune :
    - Le champ « Description » est peuplé avec une chaîne de caractères
    - Le champ « DataObjectVersion » est soit vide, soit peuplé avec un tableau d'une ou plusieurs chaînes de caractères. Chacune de ces chaînes de caractères doit correspondre à un des usages définis dans les groupe d'objets techniques pris en charge dans la solution logicielle Vitam, « BinaryMaster », « TextContent », « Thumbnail », « PhysicalMaster », « Dissemination »
    - Le champ « OriginatingAgencies » est soit vide soit peuplé avec un tableau d'une ou plusieurs chaînes de caractères. Chacune de ces chaînes de caractères doit correspondre au champ « Identifier » d'un service agent contenu dans le référentiel des services agents.
    - Le champ « WritingPermission » doit être à « true » ou « false »
    - Le champ « EveryOriginatingAgency » doit être à « true » ou « false »
    - Le champ « EveryDataObjectVersion » doit être à « true » ou « false »
    - Le champ « WritingRestrictedDesc » : droit de modification des métadonnées descriptives seulement
    - Le champ « RootUnit » est soit vide, soit peuplé avec un tableau d'une ou plusieurs chaînes de caractère. Chacune des chaînes de caractère doit correspondre au GUID d'une unité archivistique prise en charge dans la solution logicielle Vitam
- **Type** : bloquant
- **Statuts** :
  - OK : le contrat répond aux exigences des règles (STP_IMPORT_ACCESS_CONTRACT.OK = Succès du processus d'import du contrat d'accès)
  - KO : une des règles ci-dessus n'a pas été respectée (STP_IMPORT_ACCESS_CONTRACT.KO = Échec du processus d'import du contrat d'accès)

- Cas n°1 : l'identifiant du contrat est déjà utilisé (STP_IMPORT_ACCESS_CONTRACT.IDENTIFIER_DUPLICATION.KO = Échec de l'import du contrat d'accès : l'identifiant est déjà utilisé)
- Cas n°2 : au moins un des champs obligatoires n'est pas renseigné (STP_IMPORT_ACCESS_CONTRACT.EMPTY_REQUIRED_FIELD.KO = Échec de l'import du contrat d'accès : au moins un des champs obligatoires n'est pas renseigné)
- Cas n°3 : service producteur inconnu (STP_IMPORT_ACCESS_CONTRACT.AGENCY_NOT_FOUND.KO = Échec de l'import du contrat d'accès : au moins un service producteur est inconnu)
- Cas n°4 : erreur de validation du contrat (STP_IMPORT_ACCESS_CONTRACT.VALIDATION_ERROR.KO = Échec de l'import du contrat d'accès : erreur de validation du contrat du contrat d'accès)

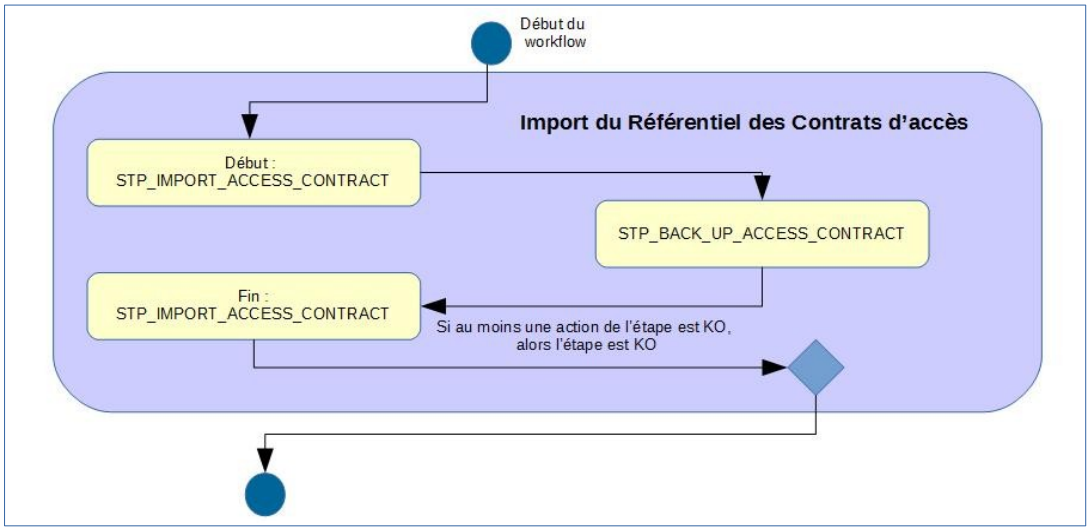
- FATAL : une erreur technique est survenue lors de la vérification de l'import du contrat d'accès (STP_IMPORT_ACCESS_CONTRACT.FATAL = Erreur technique lors du processus d'import du contrat d'accès)
- WARNING : avertissement lors du processus d'import du contrat d'accès (STP_IMPORT_ACCESS_CONTRACT.WARNING = Avertissement lors du processus d'import du contrat d'accès)

5.6.1.1. Sauvegarde du JSON STP_BACKUP_ACCESS_CONTRACT (IngestContractImpl.java)

- **Règle** : tâche consistant à enregistrer une copie de la base de données des contrats d'accès sur les offres de stockage
- **Type** : bloquant
- **Statuts** :
  - OK : une copie de la base de données nouvellement importée est enregistrée (STP_BACKUP_ACCESS_CONTRACT.OK = Succès du processus de sauvegarde des contrats d'accès)
  - KO : pas de cas KO
  - FATAL : une erreur technique est survenue lors de la copie de la base de données nouvellement importée (STP_BACKUP_ACCESS_CONTRACT.FATAL = Erreur technique lors du processus de sauvegarde des contrats d'accès)

5.6.2. Structure du Workflow d'import du référentiel des contrats d'accès

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



5.6.3. Processus de mise à jour d'un contrat d'accès STP_UPDATE_ACCESS_CONTRACT

- **Règle** : opération consistant à modifier un contrat d'accès, selon un processus identique à celui de l'import initial
- **Type** : bloquant
- **Statuts** :
  - OK : le contrat répond aux exigences des règles (STP_UPDATE_ACCESS_CONTRACT.OK = Succès du processus de mise à jour du contrat d'accès)
  - KO : une des règles ci-dessus n'a pas été respectée (STP_UPDATE_ACCESS_CONTRACT.KO = Échec du processus de mise à jour du contrat d'accès)
    - Cas n° 1 : l'identifiant du contrat d'entrée est inconnu (STP_UPDATE_ACCESS_CONTRACT.CONTRACT_NOT_FOUND.KO = Échec du processus de mise à jour du contrat d'accès : contrat d'accès non trouvé)
    - Cas n°2 : une des valeurs saisies dans le contrat d'accès ne correspond pas aux valeurs attendues (STP_UPDATE_ACCESS_CONTRACT.NOT_IN_ENUM.KO = Échec du processus de mise à jour du contrat d'accès : une valeur ne correspond pas aux valeurs attendues)
    - Cas n°3 : service producteur inconnu (STP_UPDATE_ACCESS_CONTRACT.AGENCY_NOT_FOUND.KO = Échec du processus de mise à jour du contrat d'accès : au moins un service agent est inconnu)
    - Cas n°4 : la requête de mise à jour du contrat d'accès est mal formatée (STP_UPDATE_ACCESS_CONTRACT.BAD_REQUEST.KO = Échec du processus de mise à jour du contrat d'accès : mauvaise requête)
  - FATAL : une erreur technique est survenue lors de la vérification de l'import du contrat d'accès (STP_UPDATE_ACCESS_CONTRACT.FATAL = Erreur technique lors du processus de mise à jour du contrat d'accès)

5.6.3.1. Sauvegarde du JSON STP_BACKUP_ACCESS_CONTRACT (IngestContractImpl.java)

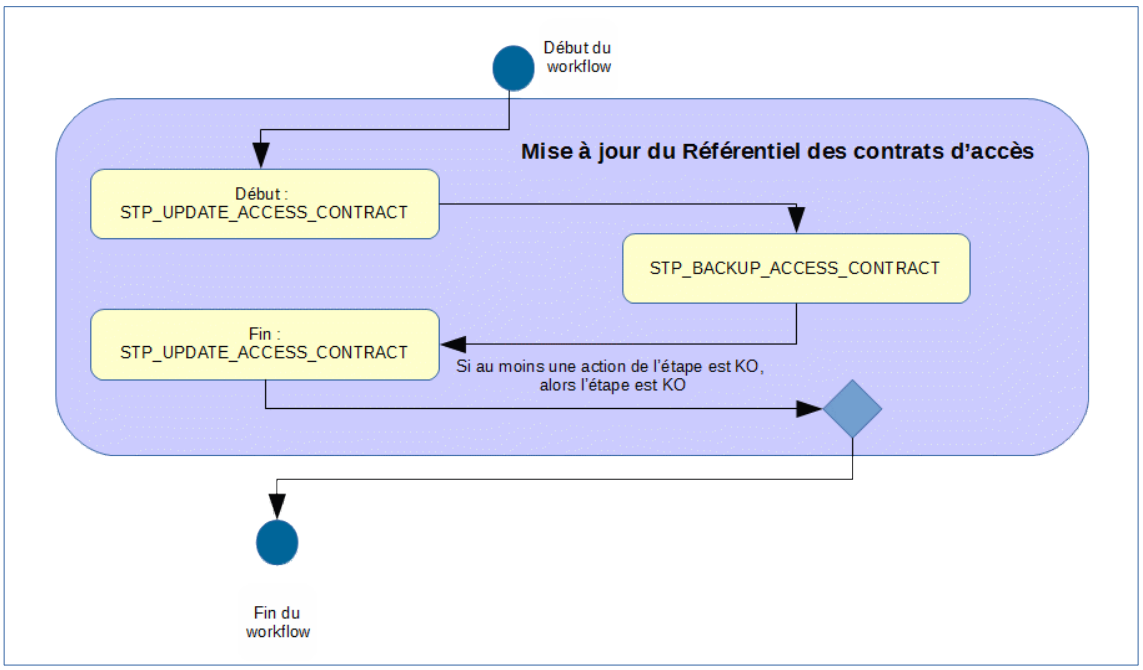
- **Règle** : tâche consistant à enregistrer une copie de la base de données des contrats d'accès sur les offres de stockage
- **Type** : bloquant
- **Statuts** :
  - OK : une copie de la base de données nouvellement importée est enregistrée (STP_BACKUP_ACCESS_CONTRACT.OK = Succès du processus de sauvegarde des contrats

d’accès)

- KO : pas de cas KO
- FATAL : une erreur technique est survenue lors de la copie de la base de données nouvellement importée (STP_BACKUP_ACCESS_CONTRACT.FATAL = Erreur technique lors du processus de sauvegarde des contrats d’accès)

5.6.4.Structure du Workflow de mise à jour d'un référentiel des contrats d'accès

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



5.7.Workflow d'administration d'un référentiel des contrats de gestion

Cette section décrit le processus (workflow) permettant d'importer et de mettre à jour un contrat de gestion.

Toutes les étapes, tâches et traitements sont journalisés dans le journal des opérations et décrivent le processus (clé et description de la clé associée dans le journal des opérations) tel qu'implémenté dans la version actuelle de la solution logicielle Vitam.

5.7.1.Processus d'import et mise à jour d'un contrat de gestion (STP_IMPORT_MANAGEMENT_CONTRACT)

Le processus d'import d'un contrat de gestion permet à la fois de vérifier qu'il contient les informations minimales obligatoires, de vérifier la cohérence de l'ensemble des informations et de lui affecter des éléments peuplés automatiquement.

Tous les éléments réalisés au cours de ce processus sont exécutés dans une seule étape.

- **Règle :** opération consistant à vérifier la présence des informations minimales dans le contrat de gestion, à s'assurer de la cohérence des informations saisies, à affecter des données aux champs peuplés par la solution logicielle Vitam et et à enregistrer le contrat
  - Les données suivantes sont obligatoirement remplies :
    - Le champ « Name » est peuplé d'une chaîne de caractères
    - Le champ « Identifier » est peuplé d'une chaîne de caractères si le référentiel des contrats de gestion est configuré en mode esclave sur le tenant sélectionné
    - Le champ « _tenant » : identifiant du tenant.
    - Le champ « Status » est peuplé avec la valeur « ACTIVE » ou la valeur « INACTIVE »
  - Les données suivantes optionnelles, si elles sont remplies, le sont en respectant les règles énoncées pour chacune :
    - Le champ « Description » est peuplé avec une chaîne de caractères
    - Le sous-objet « Storage » est soit vide, soit peuplé avec les champs suivants
      - Le champ « UnitStrategy » est soit vide soit peuplé avec un identifiant de stratégie de stockage déployé sur la plateforme VITAM et contenant l'offre de référence définie dans la stratégie de plateforme VITAM (« default »)
      - Le champ « ObjectGroupStrategy » est soit vide soit peuplé avec un identifiant de stratégie de stockage déployé sur la plateforme VITAM et contenant l'offre de référence définie dans la stratégie de plateforme VITAM (« default »)
      - Le champ « ObjectStrategy » est soit vide soit peuplé avec un identifiant de stratégie de stockage déployé sur la plateforme VITAM
- **Type** : bloquant
- **Statuts** :
  - OK : le contrat répond aux exigences des règles (STP_IMPORT_MANAGEMENT_CONTRACT.OK = Succès du processus d'import du contrat de gestion)
  - KO : une des règles ci-dessus n'a pas été respectée (STP_IMPORT_MANAGEMENT_CONTRACT.KO = Échec du processus d'import du contrat de gestion)
    - Cas n°1 : l'identifiant du contrat est déjà utilisé (STP_IMPORT_MANAGEMENT_CONTRACT.IDENTIFIER_DUPLICATION.KO = Échec de l'import du contrat de gestion : l'identifiant est déjà utilisé)
    - Cas n°2 : au moins un des champs obligatoires n'est pas renseigné (STP_IMPORT_MANAGEMENT_CONTRACT.EMPTY_REQUIRED_FIELD.KO = Échec de l'import du contrat de gestion : au moins un des champs obligatoires n'est pas renseigné)
    - Cas n°3 : une des stratégies est invalide

(STP_IMPORT_MANAGEMENT_CONTRACT.STRATEGY_VALIDATION_ERROR.KO = Échec de l'import du contrat de gestion : au moins une stratégie est invalide)

- Cas n°4 : erreur de validation du contrat (STP_IMPORT_MANAGEMENT_CONTRACT.VALIDATION_ERROR.KO = Échec de l'import du contrat de gestion : erreur de validation du contrat du contrat de gestion)

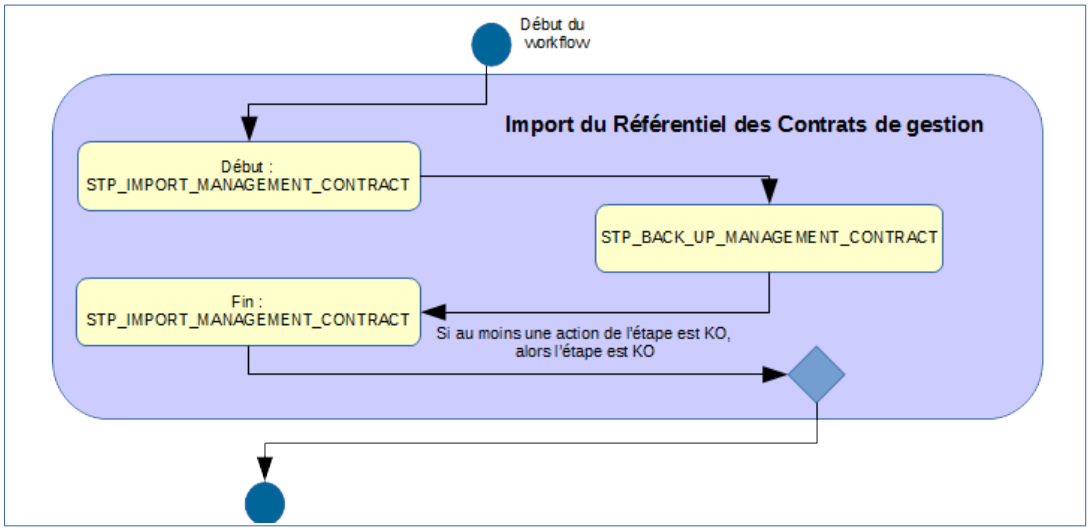
- FATAL : une erreur technique est survenue lors de la vérification de l'import du contrat de gestion (STP_IMPORT_MANAGEMENT_CONTRACT.FATAL = Erreur technique lors du processus d'import du contrat de gestion)
- WARNING : avertissement lors du processus d'import du contrat de gestion (STP_IMPORT_MANAGEMENT_CONTRACT.WARNING = Avertissement lors du processus d'import du contrat de gestion)

5.7.1.1. Sauvegarde du JSON STP_BACKUP_MANAGEMENT_CONTRACT (ManagementContractImpl.java)

- **Règle** : tâche consistant à enregistrer une copie de la base de données des contrats d'accès sur les offres de stockage
- **Type** : bloquant
- **Statuts** :
  - OK : une copie de la base de données nouvellement importée est enregistrée (STP_BACKUP_MANAGEMENT_CONTRACT.OK = Succès du processus de sauvegarde des contrats de gestion)
  - KO : pas de cas KO
  - FATAL : une erreur technique est survenue lors de la copie de la base de données nouvellement importée (STP_BACKUP_MANAGEMENT_CONTRACT.FATAL = Erreur technique lors du processus de sauvegarde des contrats de gestion)

5.7.2.Structure du Workflow d'import du référentiel des contrats de gestion

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



5.7.3. Processus de mise à jour d'un contrat de gestion (STP_UPDATE_MANAGEMENT_CONTRACT)

- **Règle** : opération consistant à modifier un contrat de gestion, selon un processus identique à celui de l'import initial
- **Type** : bloquant
- **Statuts** :
  - OK : le contrat répond aux exigences des règles (STP_UPDATE_MANAGEMENT_CONTRACT.OK = Succès du processus de mise à jour du contrat de gestion)
  - KO : une des règles ci-dessus n'a pas été respectée (STP_UPDATE_MANAGEMENT_CONTRACT.KO = Échec du processus de mise à jour du contrat de gestion)

- Cas n° 1 : l'identifiant du contrat d'entrée est inconnu (STP_UPDATE_MANAGEMENT_CONTRACT.CONTRACT_NOT_FOUND.KO = Échec du processus de mise à jour du contrat de gestion : contrat de gestion non trouvé)
- Cas n°2 : une des valeurs saisies dans le contrat de gestion ne correspond pas aux valeurs attendues (STP_UPDATE_MANAGEMENT_CONTRACT.NOT_IN_ENUM.KO = Échec du processus de mise à jour du contrat de gestion : une valeur ne correspond pas aux valeurs attendues)
- Cas n°3 : une des stratégies est invalide (STP_UPDATE_MANAGEMENT_CONTRACT.STRATEGY_VALIDATION_ERROR.KO = Échec du processus de mise à jour du contrat de gestion : erreur de validation du contrat du contrat de gestion)
- Cas n°4 : la requête de mise à jour du contrat de gestion est mal formatée (STP_UPDATE_MANAGEMENT_CONTRACT.BAD_REQUEST.KO = Échec du processus de mise à jour du contrat de gestion : mauvaise requête)

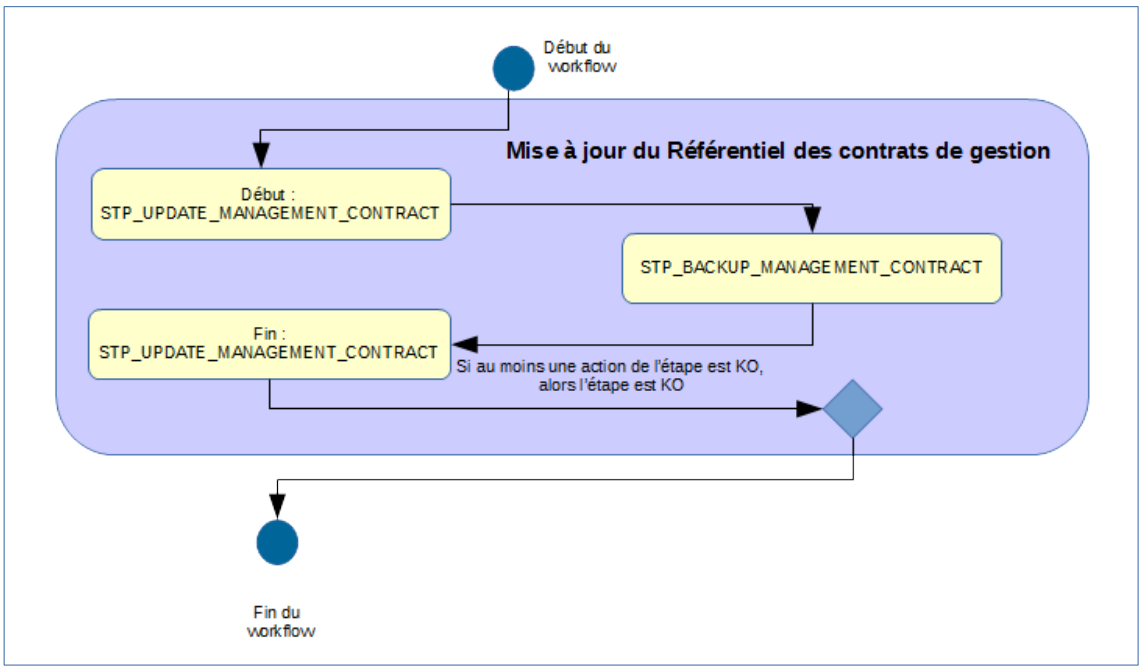
- FATAL : une erreur technique est survenue lors de la vérification de l'import du contrat de gestion (STP_UPDATE_MANAGEMENT_CONTRACT.FATAL = Erreur technique lors du processus de mise à jour du contrat de gestion)

5.7.3.1. Sauvegarde du JSON STP_BACKUP_MANAGEMENT_CONTRACT (ManagementContractImpl.java)

- **Règle** : tâche consistant à enregistrer une copie de la base de données des contrats de gestion sur les offres de stockage
- **Type** : bloquant
- **Statuts** :
  - OK : une copie de la base de données nouvellement importée est enregistrée (STP_BACKUP_MANAGEMENT_CONTRACT.OK = Succès du processus de sauvegarde des contrats de gestion)
  - KO : pas de cas KO
  - FATAL : une erreur technique est survenue lors de la copie de la base de données nouvellement importée (STP_BACKUP_MANAGEMENT_CONTRACT.FATAL = Erreur technique lors du processus de sauvegarde des contrats de gestion)

5.7.4.Structure du Workflow de mise à jour d'un référentiel des contrats de gestion

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



5.8.Workflow d'administration d'un référentiel des profils d'archivage

Cette section décrit le processus (workflow) permettant d'importer et de mettre à jour un profil d'archivage.

Toutes les étapes, tâches et traitements sont journalisées dans le journal des opérations et décrivent le processus (clé et description de la clé associée dans le journal des opérations) tel qu'implémenté dans la version actuelle de la solution logicielle Vitam.

5.8.1.Processus d'import d'une notice de profil d'archivage (STP_IMPORT_PROFILE_JSON)

Le processus d'import d'une notice de profil d'archivage permet à la fois de vérifier qu'il contient les informations minimales obligatoires, de vérifier la cohérence de l'ensemble des informations, et de lui affecter des éléments peuplés automatiquement. Tous les éléments réalisés au cours de ce processus sont exécutés dans une seule étape.

Règle : opération consistant à vérifier la présence des informations minimales dans la notice de profil d'archivage, à s'assurer de la cohérence des informations saisies, à affecter des données aux champs peuplés par la solution logicielle Vitam et à enregistrer le profil

- Les données suivantes sont obligatoirement remplies :
  - Le champ « Name » est peuplé d'une chaîne de caractères
  - Le champ « Identifier » est peuplé d'une chaîne de caractère si le référentiel des profils d'archivage est configuré en mode esclave sur le tenant séléctionné
  - Le champ « Format » doit être renseigné avec la valeur RNG ou XSD
- Les données suivantes optionnelles si elles sont remplies le sont en respectant les règles énoncées pour chacune :
  - Le champ « Description » est peuplé avec une chaîne de caractères
  - Le champ « Status » est peuplé avec la valeur ACTIVE ou la valeur INACTIVE

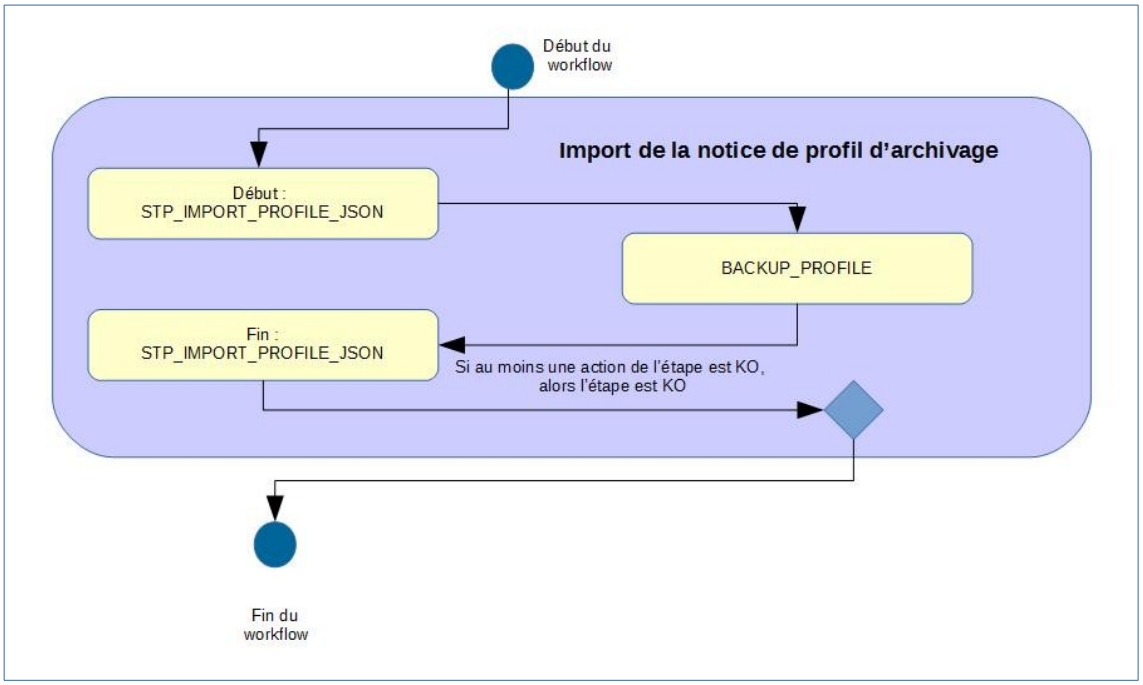
- **Type** : bloquant
- **Statuts** :
  - OK : les règles ci-dessus sont respectées (STP_IMPORT_PROFILE_JSON.OK=Succès du processus d'import du profil d'archivage)
  - KO :
    - Cas n°1 : une des règles ci-dessus n'a pas été respectée (STP_IMPORT_PROFILE_JSON.KO = Échec du processus d'import du profil d'archivage)
    - Cas n°2 : l'identifiant est déjà utilisé (STP_IMPORT_PROFILE_JSON.IDENTIFIER_DUPLICATION.KO = Échec de l'import du profil d'archivage : l'identifiant est déjà utilisé)
    - Cas n°3 : au moins un des champs obligatoires n'est pas renseigné (STP_IMPORT_PROFILE_JSON.EMPTY_REQUIRED_FIELD.KO = Échec de l'import du profil d'archivage : au moins un des champs obligatoires n'est pas renseigné)
    - Cas n°4 : le profil d'archivage est inconnu (STP_IMPORT_PROFILE_JSON.PROFILE_NOT_FOUND.KO = Échec de l'import du profil d'archivage : profil d'archivage non trouvé)
  - WARNING : avertissement lors du processus d'import du profil d'archivage (STP_IMPORT_PROFILE_JSON.WARNING = Avertissement lors du processus d'import du profil d'archivage)
  - FATAL : une erreur technique est survenue lors de la vérification de l'import du profil d'archivage (STP_IMPORT_PROFILE_JSON.FATAL = Erreur technique lors du processus d'import du profil d'archivage)

5.8.1.1. Sauvegarde du JSON BACKUP_PROFILE (ProfileServiceImpl.java)

- **Règle** : tâche consistant à enregistrer une copie de la base de données des métadonnées de profils d'archivage sur les offres de stockage
- **Type** : bloquant
- **Statuts** :
  - OK : une copie de la base de données nouvellement importée est enregistrée (BACKUP_PROFILE.OK = Succès du processus de sauvegarde des profils d'archivage)
  - KO : pas de cas KO
  - FATAL : une erreur technique est survenue lors de la copie de la base de données nouvellement importée (BACKUP_PROFILE.FATAL = Erreur technique lors du processus de sauvegarde des profils d'archivage)

5.8.2. Structure du Workflow d'import d'une notice de profil d'archivage

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



5.8.3.Processus de mise à jour d'une notice de profil d'archivage (STP_UPDATE_PROFILE_JSON)

Le processus de mise à jour d'une notice de profil d'archivage permet à la fois de vérifier qu'il contient les informations minimales obligatoires, de vérifier la cohérence de l'ensemble des informations, et de lui affecter des éléments peuplés automatiquement.

Tous les éléments réalisés au cours de ce processus sont exécutés dans une seule étape.

- **Règle** : opération consistant à mettre à jour la notice d'un profil d'archivage, selon les mêmes règles que celles décrites pour la création.

  L'association d'un fichier de profil avec les métadonnées d'un profil provoque également une opération de mise à jour du profil d'archivage.
- **Type** : bloquant
- **Statuts** :
  - OK : le fichier importé est au même format que celui décrit dans le champ « Format » (STP_UPDATE_PROFILE_JSON.OK = Succès du processus du processus de mise à jour du profil d'archivage (fichier xsd ou rng)
  - KO :
    - Cas n°1 : le fichier importé n'est pas au même format que celui décrit dans le champ « Format » (STP_UPDATE_PROFILE_JSON.KO = Échec du processus de mise à jour du profil d'archivage (fichier xsd ou rng)
    - Cas n°2 : le profil d'archivage est inconnu (STP_UPDATE_PROFILE_JSON.PROFILE_NOT_FOUND.KO = Échec du processus de mise à jour du profil d'archivage : profil non trouvé)
    - Cas n°3 : une des valeurs saisies dans le profil d'archivage ne correspond pas aux valeurs attendues (STP_UPDATE_PROFILE_JSON.NOT_IN_ENUM.KO = Échec du processus de mise à jour du profil d'archivage : une valeur ne correspond pas aux valeurs attendues)
    - Cas n°4 : l'identifiant du profil d'archivage est déjà utilisé (STP_UPDATE_PROFILE_JSON.IDENTIFIER_DUPLICATION.KO = Échec du processus de mise à jour du profil d'archivage : l'identifiant est déjà utilisé
  - FATAL : une erreur technique est survenue lors de la vérification de l'import du profil d'archivage (STP_UPDATE_PROFILE_JSON.FATAL = Erreur technique lors du processus de mise à jour du profil d'archivage (fichier xsd ou rng))

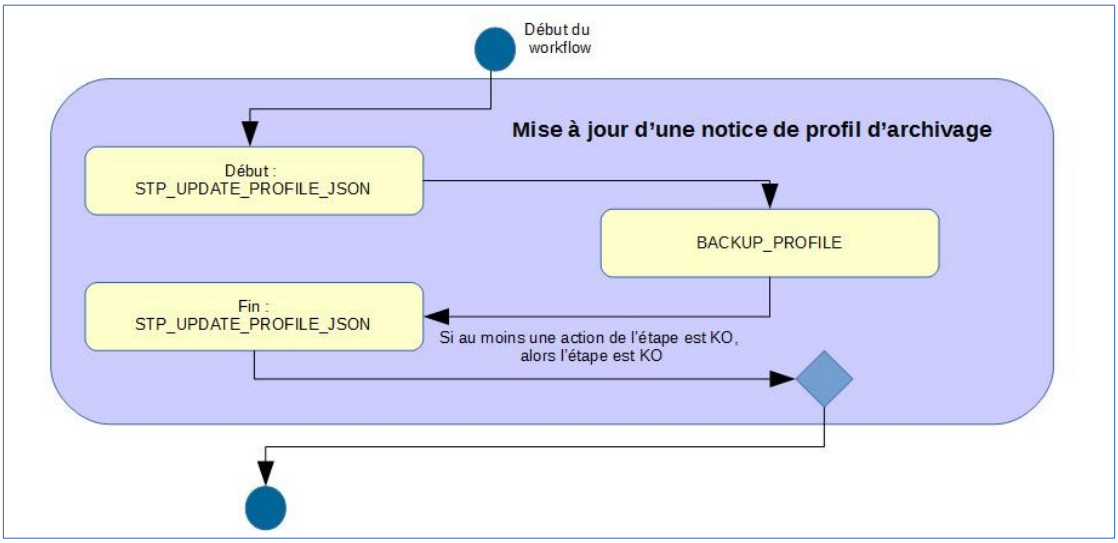
5.8.3.1. Sauvegarde du JSON BACKUP_PROFILE (ProfileServiceImpl.java)

Cette tâche est appelée que ce soit en import initial ou lors de la modification des métadonnées de profils

- **Règle** : tâche consistant à enregistrer enregistrement d'une copie de la base de données des métadonnées de profils d'archivage sur les offres de stockage
- **Type** : bloquant
- **Statuts** :
  - OK : une copie de la base de données nouvellement importée est enregistrée (BACKUP_PROFILE.OK = Succès du processus de sauvegarde des profils d'archivage)
  - KO : pas de cas KO
  - FATAL : une erreur technique est survenue lors de la copie de la base de données nouvellement importée (BACKUP_PROFILE.FATAL = Erreur technique lors du processus de sauvegarde des profils d'archivage)

5.8.4.Structure du Workflow de mise à jour d'une notice de profil d'archivage

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



5.9.Workflow d'administration d'un référentiel des profils de sécurité

Cette section décrit le processus (workflow) de création et de mise à jour d'un profil de sécurité. Cette opération n'est réalisable que sur le tenant d'administration.

Toutes les étapes, tâches et traitements sont journalisées dans le journal des opérations et décrivent le processus (clé et description de la clé associée dans le journal des opérations) tel qu'implémenté dans la version actuelle de la solution logicielle Vitam.

5.9.1.Processus d'import d'un référentiel des profils de sécurité (STP_IMPORT_SECURITY_PROFILE)

Le processus d'import d'un profil de sécurité permet à la fois de vérifier qu'il contient les informations minimales obligatoires, de vérifier la cohérence de l'ensemble des informations, et de lui affecter des éléments peuplés automatiquement.

Tous les éléments réalisés au cours de ce processus sont exécutés dans une seule étape.

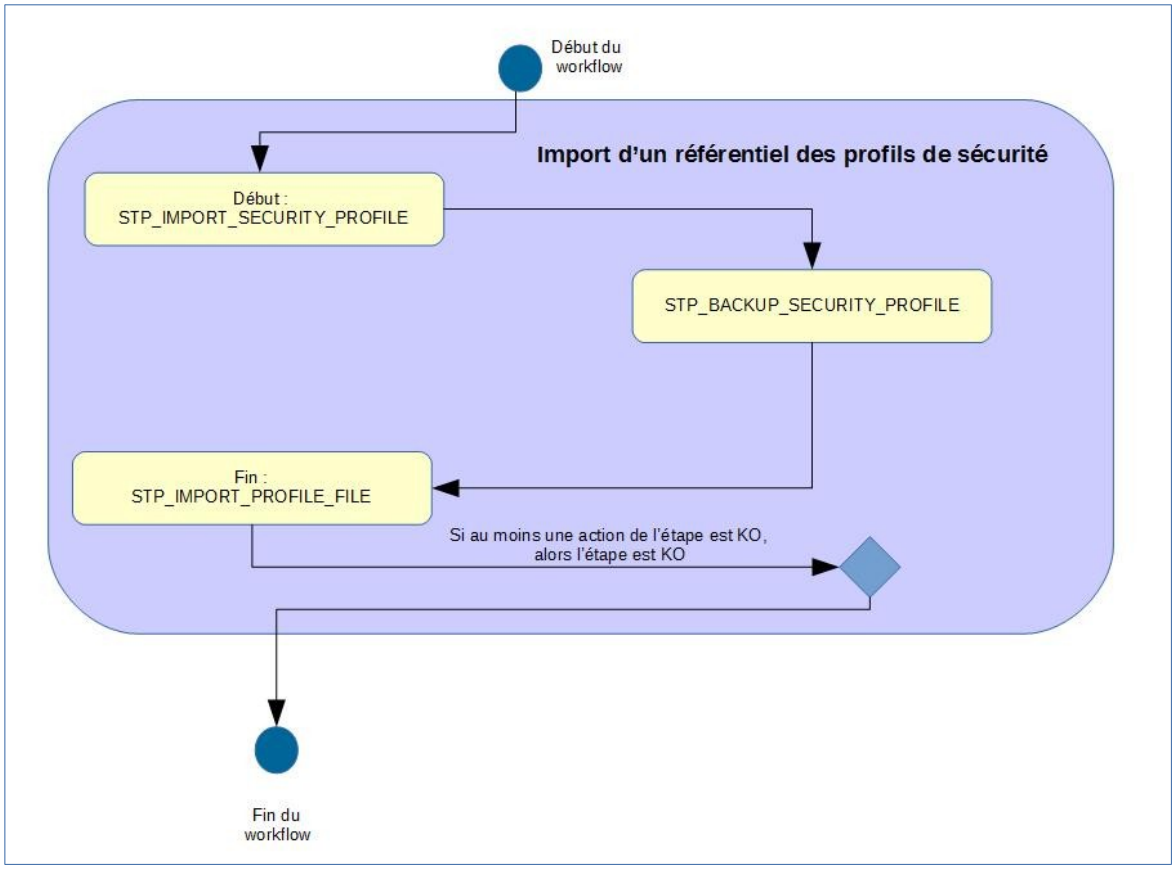
- **Règle** : étape consistant à vérifier la présence des informations minimales dans le profil de sécurité, à s'assurer de la cohérence des informations saisies, à affecter des données aux champs peuplés par la solution logicielle Vitam et et à enregistrer le profil de sécurité. Les données suivantes sont obligatoirement remplies :
  - Le champ « Name » doit être peuplé avec une chaîne de caractères unique
  - Le champ « Identifier » doit être unique
  - Le champ « FullAccess » doit être à « true » ou « false »
- **Type** : bloquant
- **Statuts** :
  - OK : les règles ci-dessus sont respectées (STP_IMPORT_SECURITY_PROFILE.OK = Succès du processus d'import du profil de sécurité)
  - KO : une des règles ci-dessus n'est pas respectée (STP_IMPORT_SECURITY_PROFILE.KO = Échec du processus d'import du profil de sécurité)
  - FATAL : une erreur technique est survenue lors de l'import du profil de sécurité (STP_IMPORT_SECURITY_PROFILE.FATAL = Erreur technique lors du processus d'import du profil de sécurité)

5.9.1.1. Sauvegarde du JSON STP_BACKUP_SECURITY_PROFILE (SecurityProfileService.java)

- **Règle** : tâche consistant à enregistrer une copie de la base de données des profils de sécurité sur le stockage
- **Type** : bloquant
- **Statuts** :
  - OK : une copie de la base de données nouvellement importée est enregistrée (STP_BACKUP_SECURITY_PROFILE.OK = Succès du processus de sauvegarde des profils de sécurité)
  - KO : pas de cas KO
  - FATAL : une erreur technique est survenue lors de la copie de la base de données nouvellement importée (STP_BACKUP_SECURITY_PROFILE.FATAL = Erreur technique lors du processus de sauvegarde des profils de sécurité)

5.9.2. Structure du Workflow d'import d'un référentiel des profils de sécurité

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



5.9.3. Processus de mise à jour d'un référentiel des profils de sécurité (STP_UPDATE_SECURITY_PROFILE)

Le processus de mise à jour d'un profil de sécurité permet à la fois de vérifier qu'il contient les informations minimales obligatoires, de vérifier la cohérence de l'ensemble des informations, et de lui affecter des éléments peuplés automatiquement.

Tous les éléments réalisés au cours de ce processus sont exécutés dans une seule étape.

- **Règle** : opération consistant à mettre à jour le profil de sécurité, selon les mêmes règles que celles décrites pour la création
- **Type** : bloquant
- **Statuts** :
  - OK : les règles ci-dessus sont respectées (STP_UPDATE_SECURITY_PROFILE.OK = Succès du processus de mise à jour du profil de sécurité)
  - KO : une des règles ci-dessus n'est pas respectée (STP_UPDATE_SECURITY_PROFILE.KO = Échec du processus de mise à jour du profil de sécurité)
  - FATAL : une erreur technique est survenue lors de l'import du profil de sécurité (STP_UPDATE_SECURITY_PROFILE.FATAL = Erreur technique lors du processus de mise à jour du profil de sécurité)

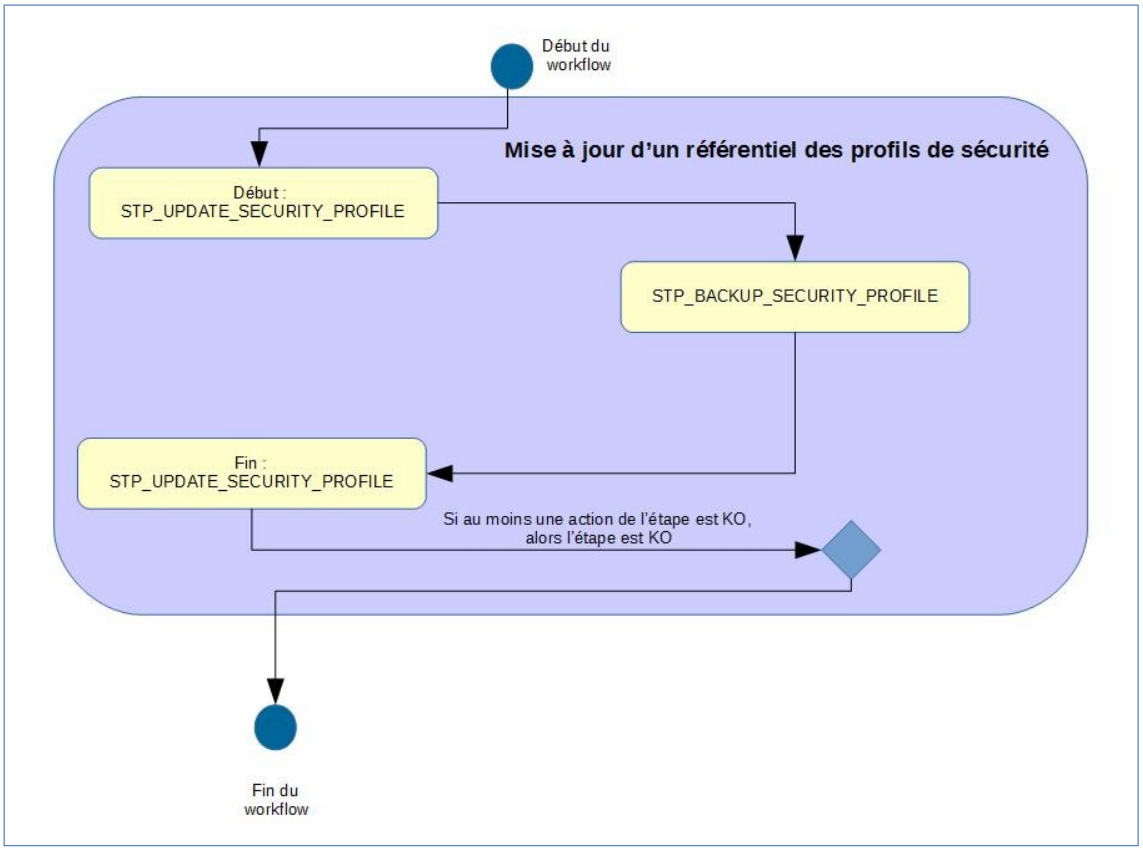
5.9.3.1. Sauvegarde du JSON STP_BACKUP_SECURITY_PROFILE (SecurityProfileService.java)

Cette tâche est appelée que ce soit en import initial ou en modification.

- **Règle** : tâche consistant à enregistrer une copie de la base de données des profils de sécurité sur le stockage
- **Type** : bloquant
- **Statuts** :
  - OK : une copie de la base de données nouvellement importée est enregistrée (STP_BACKUP_SECURITY_PROFILE.OK = Succès du processus de sauvegarde des profils de sécurité)
  - KO : pas de cas KO
  - FATAL : une erreur technique est survenue lors de la copie de la base de données nouvellement importée (STP_BACKUP_SECURITY_PROFILE.FATAL = Erreur technique lors du processus de sauvegarde des profils de sécurité)

5.9.4. Structure du Workflow de mise à jour du référentiel des profils de sécurité

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


Début du workflow
Mise à jour d'un référentiel des profils de sécurité
Début : STP_UPDATE_SECURITY_PROFILE
STP_BACKUP_SECURITY_PROFILE
Fin : STP_UPDATE_SECURITY_PROFILE
Si au moins une action de l'étape est KO, alors l'étape est KO
Fin du workflow

5.10.Workflow d'administration d'un référentiel des contextes applicatifs

Cette section décrit le processus (workflow) d'import et de mise à jour du référentiel des contextes. Cette opération n'est réalisable que sur le tenant d'administration.

Toutes les étapes, tâches et traitements sont journalisés dans le journal des opérations et décrivent le processus (clé et description de la clé associée dans le journal des opérations) tel qu'implémenté dans la version actuelle de la solution logicielle Vitam.

5.10.1.Processus d'import d'un référentiel des contextes applicatifs (STP_IMPORT_CONTEXT)

Le processus d'import d'un référentiel des contextes applicatifs permet à la fois de vérifier qu'il contient les informations minimales obligatoires, de vérifier la cohérence de l'ensemble des informations, et de lui affecter des éléments peuplés automatiquement.

Tous les éléments réalisés au cours de ce processus sont exécutés dans une seule étape.

- **Règle** : opération consistant à vérifier la présence des informations minimales dans le contexte applicatif, à s'assurer de la cohérence des informations saisies, à affecter des données aux champs peuplés par la solution logicielle Vitam et et à enregistrer le contexte applicatif :
    - Le champ « Name » doit être peuplé avec une chaîne de caractères unique
    - Le champ « Status » doit être à « ACTIVE » ou « INACTIVE »
    - Le champ « EnableControl » doit être à « true » ou « false »
    - Le champ « Permissions » doit être peuplé avec un tableau contenant des fichiers JSON
    - Le champ « SecurityProfile » doit être peuplé avec une chaîne de caractères
    - Le champ « Identifier » doit être unique
- **Type** : bloquant
- **Statuts** :
    - OK : Les règles ci-dessus sont respectées (STP_IMPORT_CONTEXT.OK = Succès du processus d'import du contexte)
    - KO :

- Cas n°1 : une des règles ci-dessus n'est pas respectée (STP_IMPORT_CONTEXT.KO = Échec du processus d'import du contexte)
- Cas n°2 : l'identifiant est déjà utilisé (STP_IMPORT_CONTEXT.IDENTIFIER_DUPLICATION.KO = Échec de l'import : l'identifiant est déjà utilisé)
- Cas n°3 : un des champs obligatoires n'est pas renseigné (STP_IMPORT_CONTEXT.EMPTY_REQUIRED_FIELD.KO = Échec de l'import : au moins un des champs obligatoires n'est pas renseigné)
- Cas n°4 : le profil de sécurité mentionné est inconnu du système (STP_IMPORT_CONTEXT.SECURITY_PROFILE_NOT_FOUND.KO = Échec de l'import : profil de sécurité non trouvé)
- Cas n°5 : au moins un objet déclare une valeur inconnue (STP_IMPORT_CONTEXT.UNKNOWN_VALUE.KO = Échec de l'import : au moins un objet déclare une valeur inconnue)

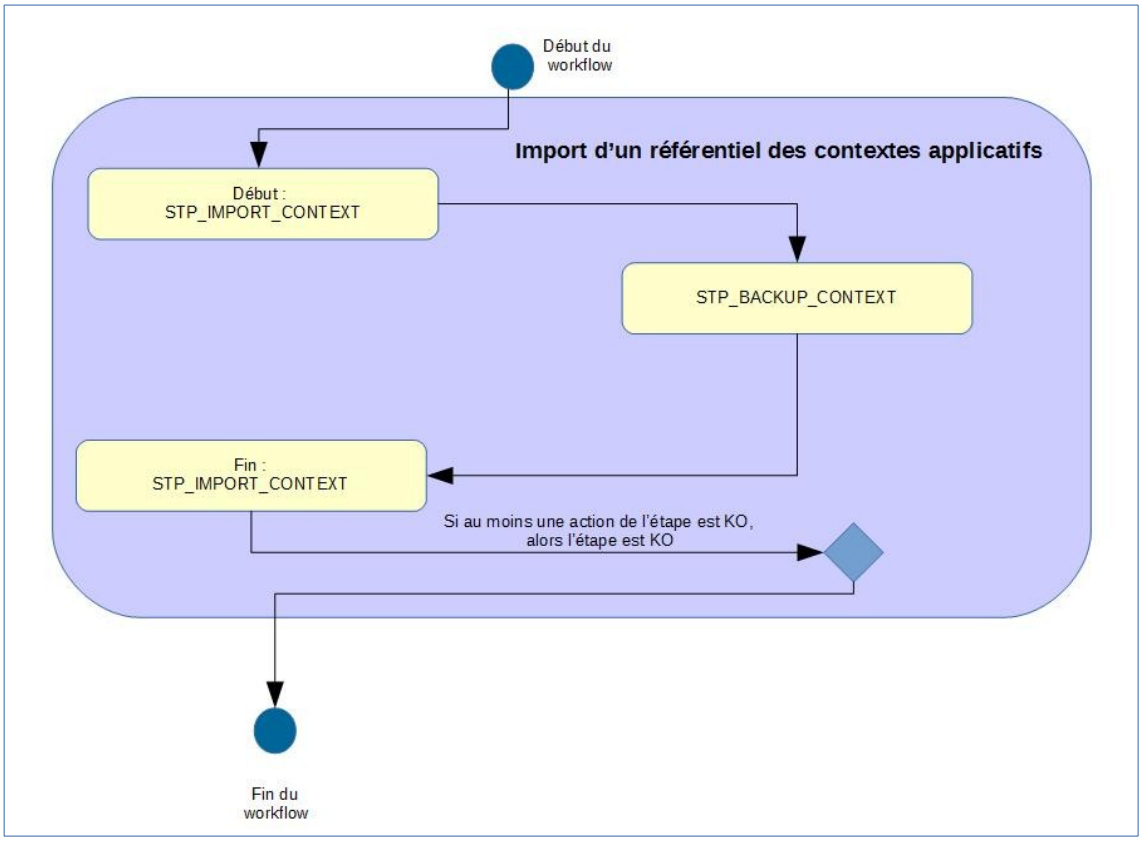
  - FATAL : une erreur technique est survenue lors de l'import du contexte (STP_IMPORT_CONTEXT.FATAL=Erreur technique lors du processus d'import du contexte)

5.10.1.1. Sauvegarde du JSON STP_BACKUP_CONTEXT (ContextServiceImpl.java)

- **Règle** : tâche consistant à enregistrer une copie de la base de données des contextes applicatifs sur le stockage
- **Type** : bloquant
- **Statuts** :
  - OK : une copie de la base de données nouvellement importée est enregistrée (STP_BACKUP_CONTEXT.OK = Succès du processus de sauvegarde des contextes)
  - KO : pas de cas KO
  - FATAL : une erreur technique est survenue lors de la copie de la base de données nouvellement importée (STP_BACKUP_CONTEXT.FATAL = Erreur technique lors du processus de sauvegarde des contextes)

5.10.2.Structure du Workflow d'import d'un référentiel des contextes applicatifs

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



5.10.3. Processus de mise à jour d'un référentiel des contextes applicatifs (STP_UPDATE_CONTEXT)

Le processus de mise à jour d'un contexte applicatif permet à la fois de vérifier qu'il contient les informations minimales obligatoires, de vérifier la cohérence de l'ensemble des informations, et de lui affecter des éléments peuplés automatiquement.

Tous les éléments réalisés au cours de ce processus sont exécutés dans une seule étape.

- **Règle** : opération consistant à mettre à jour le contexte applicatif, selon les mêmes règles que celles décrites pour la création
- **Type** : bloquant
- **Statuts** :
  - OK : les règles ci-dessus sont respectées (STP_UPDATE_CONTEXT.OK = Succès du processus de mise à jour du contexte)
  - KO : une des règles ci-dessus n'est pas respectée (STP_UPDATE_CONTEXT.KO = Échec du processus mise à jour du contexte)
    - Cas n°1 : une des valeurs saisies dans le contexte applicatif ne correspond pas aux valeurs attendues (STP_UPDATE_CONTEXT.UNKNOWN_VALUE.KO = Échec du processus de mise à jour du contexte : au moins un objet déclare une valeur inconnue)
  - FATAL : une erreur technique est survenue lors de l'import du contexte (STP_UPDATE_CONTEXT.FATAL = Erreur technique lors du processus de mise à jour du contexte)

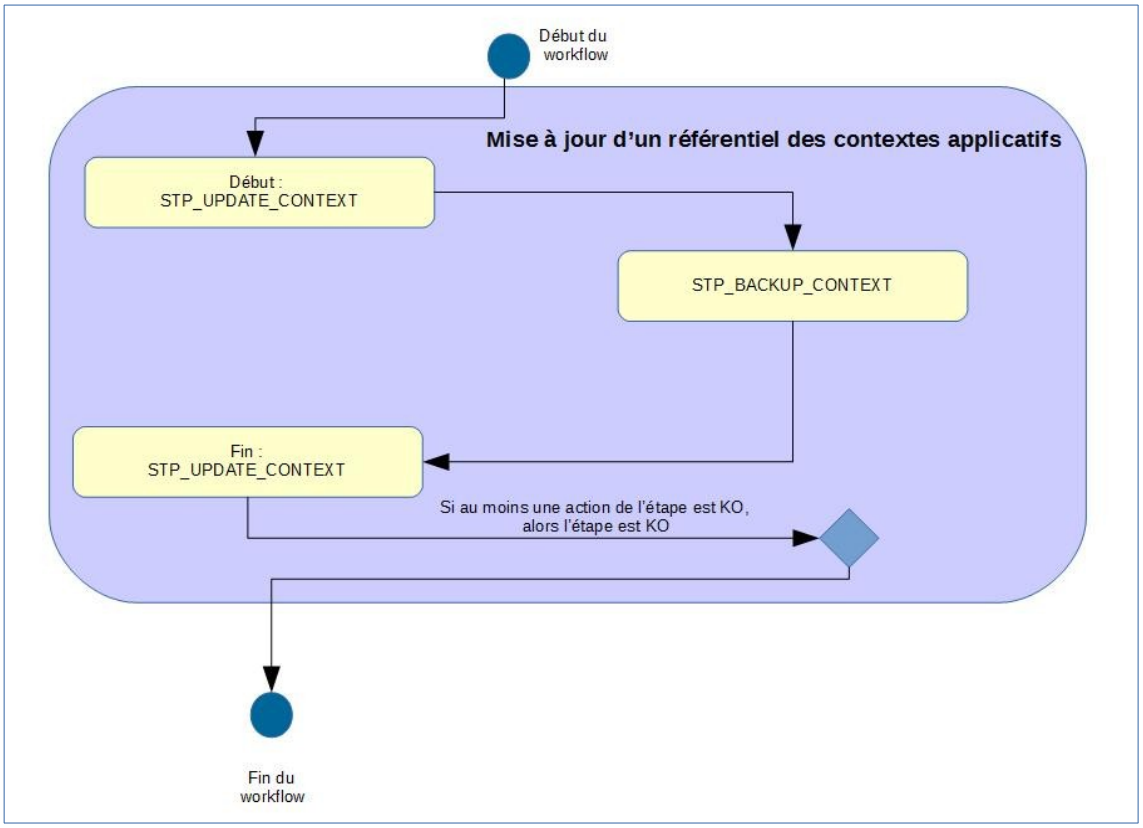
5.10.3.1. Sauvegarde du JSON STP_BACKUP_CONTEXT (ContextServiceImpl.java)

Cette tâche est appelée que ce soit en import initial ou en modification.

- **Règle** : enregistrement d'une copie de la base de données des contextes applicatifs sur le stockage
- **Type** : bloquant
- **Statuts** :
  - OK : une copie de la base de données nouvellement importée est enregistrée (STP_BACKUP_CONTEXT.OK = Succès du processus de sauvegarde des contextes)
  - KO : pas de cas KO
  - FATAL : une erreur technique est survenue lors de la copie de la base de données nouvellement importée (STP_BACKUP_CONTEXT.FATAL = Erreur technique lors du processus de sauvegarde des contextes)

5.10.4.Structure du Workflow de mise à jour du référentiel des contextes applicatifs

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


Début du workflow
Mise à jour d'un référentiel des contextes applicatifs
Début : STP_UPDATE_CONTEXT
STP_BACKUP_CONTEXT
Fin : STP_UPDATE_CONTEXT
Si au moins une action de l'étape est KO, alors l'étape est KO
Fin du workflow

5.11.Workflow d'administration du référentiel des profils d'unités archivistiques

Cette section décrit le processus (workflow) permettant d'importer et de mettre à jour un profil d'unité archivistique (documents type).

Toutes les étapes, tâches et traitements sont journalisés dans le journal des opérations et décrivent le processus (clé et description de la clé associée dans le journal des opérations) tel qu'implémenté dans la version actuelle de la solution logicielle Vitam.

5.11.1.Processus d'import d'un référentiel des profils d'unité archivistique (IMPORT_ARCHIVEUNITPROFILE)

Le processus d'import d'un profil d'unité archivistique (document type) permet à la fois de vérifier qu'il contient les informations minimales obligatoires, de vérifier la cohérence de l'ensemble des informations, et de lui affecter des éléments peuplés automatiquement.

Tous les éléments réalisés au cours de ce processus sont exécutés dans une seule étape.

- **Règle** : opération consistant à vérifier la présence des informations minimales dans le profil d'unité archivistique, à s'assurer de la cohérence des informations saisies, à affecter des données aux champs peuplés par la solution logicielle Vitam et à enregistrer le profil d'unité archivistique :

  Les données suivantes sont obligatoirement remplies :

  - Le champ « Name » est peuplé d'une chaîne de caractères
  - Le champ « Identifier » est peuplé d'une chaîne de caractères si le référentiel des profils d'unité archivistique est configuré en mode esclave sur le tenant sélectionné

  Les données suivantes, optionnelles, si elles sont remplies, le sont en respectant les règles énoncées pour chacune :

  - Le champ « Description » est peuplé avec une chaîne de caractères
  - Le champ « Status » est peuplé avec la valeur « ACTIVE » ou la valeur « INACTIVE »

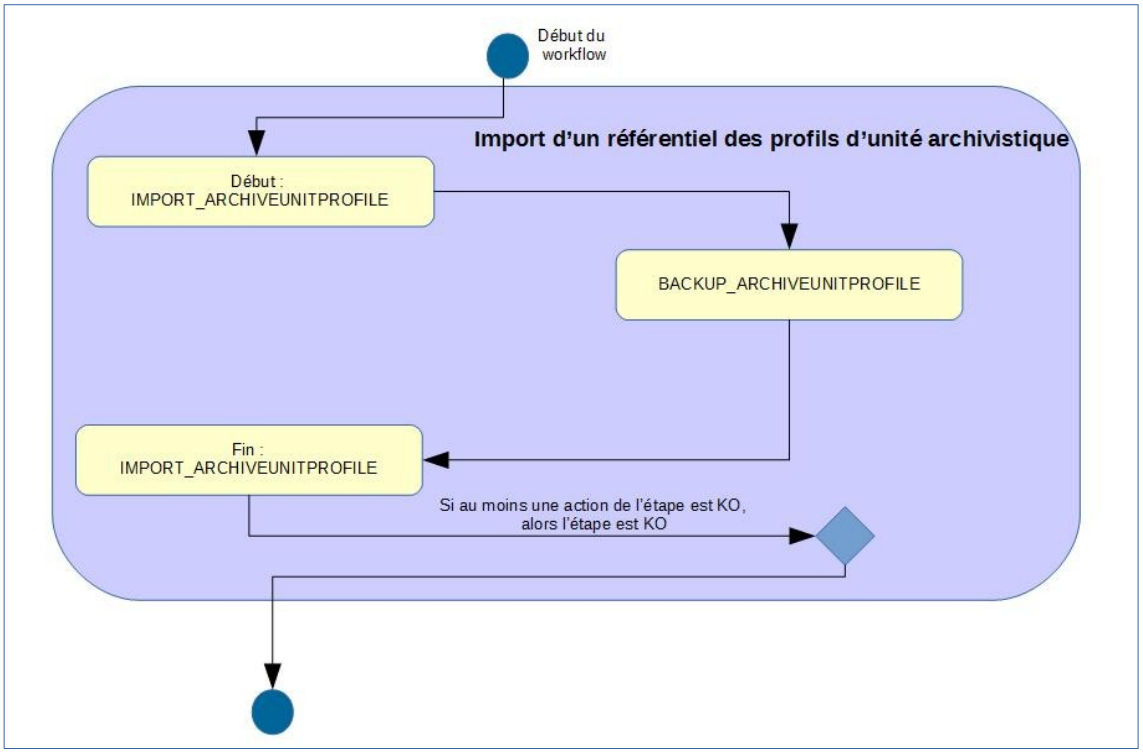
- **Type** : bloquant
- **Statuts** :
- OK : les règles ci-dessus sont respectées (IMPORT_ARCHIVEUNITPROFILE.OK = Succès du processus d'import du profil d'unité archivistique)
- KO :
  - Cas n°1 : une des règles ci-dessus n'a pas été respectée (IMPORT_ARCHIVEUNITPROFILE.KO = Échec du processus d'import du profil d'unité archivistique
  - Cas n°2 : l'identifiant est déjà utilisé (IMPORT_ARCHIVEUNITPROFILE.IDENTIFIER_DUPLICATION.KO = Échec de l'import du profil d'unité archivistique : l'identifiant est déjà utilisé)
  - Cas n°3 : au moins un des champs obligatoires n'est pas renseigné (IMPORT_ARCHIVEUNITPROFILE.EMPTY_REQUIRED_FIELD.KO = Échec de l'import du profil d'unité archivistique : au moins un des champs obligatoires n'est pas renseigné)
  - Cas n°4 : Schéma JSON invalide (IMPORT_ARCHIVEUNITPROFILE.INVALID_JSON_SCHEMA.KO = Échec de l'import du profil d'unité archivistique : schéma JSON non valide)
- FATAL : une erreur technique est survenue lors de la vérification de l'import du profil d'unité archivistique (document type) (IMPORT_ARCHIVEUNITPROFILE.FATAL = Erreur technique lors du processus d'import du profil d'unité archivistique (document type)

5.11.1.1. Sauvegarde du JSON STP_BACKUP_ARCHIVEUNITPROFILE (ArchiveUnitProfileManager.java)

- **Règle** : tâche consistant à enregistrer une copie de la base de données des profils d'unités archivistiques sur les offres de stockage
- **Type** : bloquant
- **Statuts** :
  - OK : une copie de la base de données nouvellement importée est enregistrée (BACKUP_ARCHIVEUNITPROFILE.OK = Succès du processus de sauvegarde des profils d'unité archivistique (document type)
  - KO : pas de cas KO
  - FATAL : une erreur technique est survenue lors de la copie de la base de données des profils d'unités archivistiques (document type) (BACKUP_ARCHIVEUNITPROFILE.FATAL = Erreur technique lors du processus de sauvegarde des profils d'unité archivistique (document type)

5.11.2.Structure du Workflow d'import d'un référentiel des profils d'unité archivistique

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



5.11.3.Processus de mise à jour d'un profil d'unité archivistique (UPDATE_ARCHIVEUNITPROFILE)

Le processus d'import d'un profil d'unité archivistique (document type) permet à la fois de vérifier qu'il contient les informations minimales obligatoires, de vérifier la cohérence de l'ensemble des informations, et de lui affecter des éléments peuplés automatiquement.

Tous les éléments réalisés au cours de ce processus sont exécutés dans une seule étape.

- **Règle** : opération consistant à mettre à jour le profil d'unité archivistique, selon les mêmes règles que celles décrites pour la création
- **Type** : bloquant
- **Statuts** :
  - OK : les règles ci-dessus sont respectées (UPDATE_ARCHIVEUNITPROFILE.OK = Succès du processus de mise à jour du profil d'unité archivistique (document type)
  - KO : une des règles ci-dessus n'a pas été respectée (UPDATE_ARCHIVEUNITPROFILE.KO = Échec du processus d'import du profil d'unité archivistique (document type)
    - Cas n°1 : l'identifiant du profil d'unité archivistique est inconnu (UPDATE_ARCHIVEUNITPROFILE.AUP_NOT_FOUND.KO = Échec du processus de mise à jour du profil d'unité archivistique : profil d'unité archivistique non trouvé)
    - Cas n°2 : une des valeurs saisies dans le profil d'unité archivistique ne correspond pas aux valeurs attendues (UPDATE_ARCHIVEUNITPROFILE.NOT_IN_ENUM.KO = Échec du processus de mise à jour du profil d'unité archivistique : une valeur ne correspond pas aux valeurs attendues)
    - Cas n°3 : l'identifiant est déjà utilisé (UPDATE_ARCHIVEUNITPROFILE.IDENTIFIER_DUPLICATION.KO = Échec du processus de mise à jour du profil d'unité archivistique : l'identifiant est déjà utilisé)
  - FATAL : une erreur technique est survenue lors du processus de mise à jour du profil d'unité

archivistique (UPDATE_ARCHIVEUNITPROFILE.FATAL = Erreur technique lors du processus de mise à jour du profil d'unité archivistique (document type)

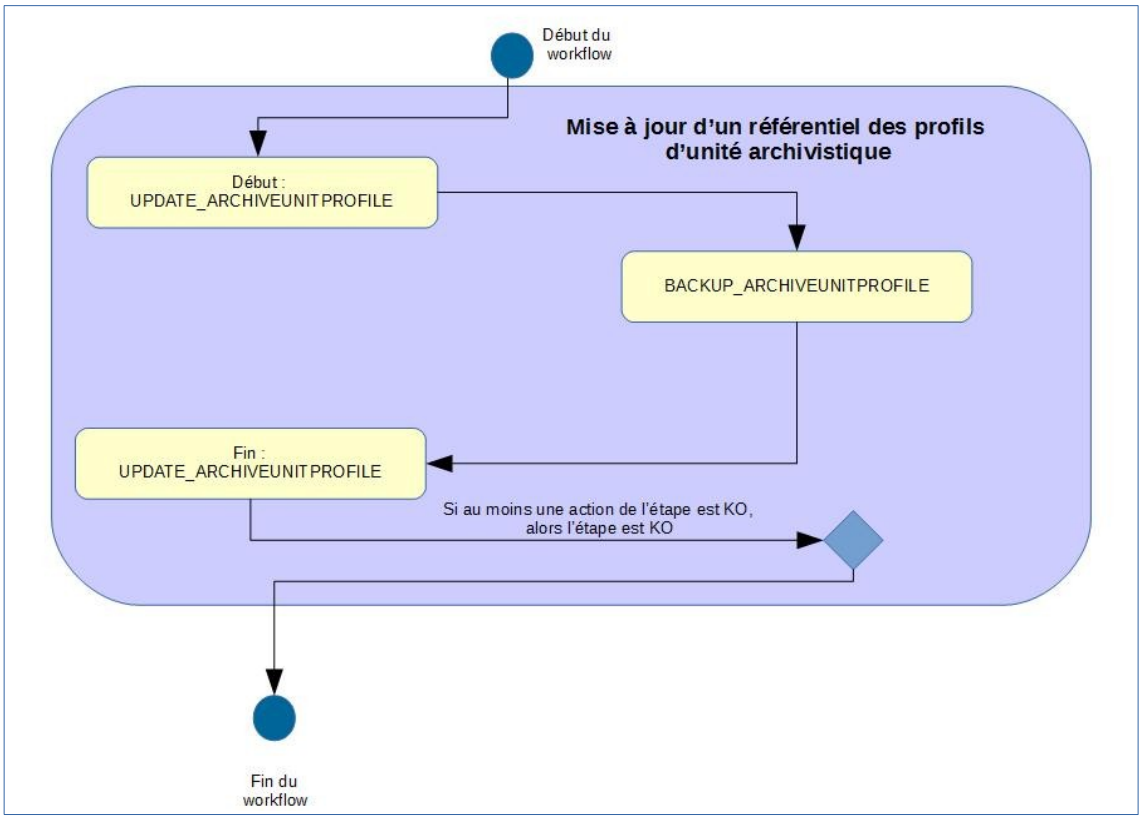
5.11.3.1. Sauvegarde du JSON STP_BACKUP_ARCHIVEUNITPROFILE (ArchiveUnitProfileManager.java)

Cette tâche est appelée que ce soit en import initial ou lors de la modification des métadonnées de profil d'unité archivistique (document type).

- **Règle** : tâche consistant à enregistrer un enregistrement d'une copie de la base de données des métadonnées des profils d'unités archivistiques sur le stockage les offres de stockage
- **Type** : bloquant
- **Statuts** :
  - OK : une copie de la base de données nouvellement importée est enregistrée (BACKUP_ARCHIVEUNITPROFILE.OK = Succès du processus de sauvegarde des profils d'unité archivistique (document type)
  - KO : pas de cas KO
  - FATAL : une erreur technique est survenue lors de la copie de la base de données échec du processus de sauvegarde dudes profils d'unités archivistiques (document type) (BACKUP_ARCHIVEUNITPROFILE.KOFATAL = Erreur technique lors du processus de sauvegarde des profils d'unité archivistique (document type)

5.11.4. Structure du Workflow de mise à jour du référentiel des profils d'unité archivistique

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



5.12.Workflow d'administration d'un référentiel des vocabulaires de l'ontologie

Cette section décrit le processus permettant d'importer des vocabulaires de l'ontologie. Cette opération n'est réalisable que sur le tenant d'administration.

Toutes les étapes, tâches et traitements sont journalisés dans le journal des opérations et décrivent le processus (clé et description de la clé associée dans le journal des opérations) tel qu'implémenté dans la version actuelle de la solution logicielle Vitam.

5.12.1.Processus d'import et mise à jour des vocabulaires de l'ontologie (STP_IMPORT_ONTOLOGY)

Le processus d'import d'une ontologie permet d'ajouter des vocabulaires qui seront utilisés dans les profils d'unité archivistique (documents types) Tous les éléments réalisés au cours de ce processus sont exécutés dans une seule étape.

- **Règle** : opération consistant à vérifier la présence des informations minimales dans le référentiel des vocabulaires de l'ontologie, à s'assurer de la cohérence des informations saisies, à affecter des données aux champs peuplés par la solution logicielle Vitam et à enregistrer le référentiel des vocabulaires de l'ontologie.

  L'ontologie répond aux exigences suivantes :
  - Le fichier est au format Json.
  - Les données suivantes sont obligatoires :
    - Le champ « Identifier » est peuplé d'une chaîne de caractères
    - Le champ « Type » est peuplé par une valeur comprise dans la liste :
      - TEXT
      - KEYWORD
      - DATE
      - LONG
      - DOUBLE
      - BOOLEAN
      - GEO_POINT
      - ENUM
    - Le champ « Origin » est peuplé par la valeur « EXTERNAL » ou « INTERNAL ». L'INTERNAL correspond à l'ontologie interne de la solution logicielle Vitam
  - Les données suivantes sont facultatives; si elles sont remplies, elles respectent les règles énoncées pour chacune :
    - Le champ « SedaField » est peuplé d'une chaîne de caractères
    - Le champ « ApiField » est peuplé d'une chaîne de caractères
    - Le champ « Description » est peuplé d'une chaîne de caractères
    - Le champ « ShortName » correspond au champ traduction, il est peuplé par une chaîne de valeur
    - Le champ « Collections » indique la collection dans laquelle le vocabulaire est rattaché, ex : « Unit »

      Exemple d'ontologie :

```
[ {
"Identifier" : "AcquiredDate",
"SedaField" : "AcquiredDate",
"ApiField" : "AcquiredDate",
"Description" : "unit-es-mapping.json",
"Type" : "DATE",
"Origin" : "INTERNAL",
"ShortName" : "AcquiredDate",
```

```
"Collections" : [ "Unit" ]
}, {
"Identifier" : "BirthDate",
"SedaField" : "BirthDate",
"ApiField" : "BirthDate",
"Description" : "unit-es-mapping.json",
"Type" : "DATE",
"Origin" : "INTERNAL",
"ShortName" : "BirthDate",
"Collections" : [ "Unit" ]
}]
```

- **Type** : bloquant
- **Statuts** :
  - OK : les règles ci-dessus sont respectées (IMPORT_ONTOLOGY.OK = Succès du processus d'import de l'ontologie)
  - KO : une des règles ci-dessus n'a pas été respectée (IMPORT_ONTOLOGY.KO = Échec du processus d'import de l'ontologie)
  - FATAL : une erreur technique est survenue lors de la vérification de l'import de l'ontologie (IMPORT_ONTOLOGY.FATAL = Erreur technique lors du processus d'import de l'ontologie)
  - WARNING : avertissement lors du processus d'import de l'ontologie (IMPORT_ONTOLOGY.WARNING = Avertissement lors du processus d'import de l'ontologie)

La modification d'une ontologie s'effectue par ré-import du fichier JSON. Le nouvel import annule et remplace l'ontologie précédente. Ce ré-import observe les règles décrites dans le processus d'import, décrit plus haut.

> ***Note :*** *la mise à jour des vocabulaires de l'ontologie doit respecter certaines règles de compatibilité concernant la valeur du « Type » :*
>
> - *Le champ Type TEXT peut être modifié en KEYWORD, TEXT*
> - *Le champ Type KEYWORD peut être modifié en KEYWORD, TEXT*
> - *Le champ Type DATE peut être modifié en KEYWORD, TEXT*
> - *Le champ Type LONG peut être modifié en KEYWORD, TEXT, DOUBLE*
> - *Le champ Type DOUBLE peut être modifié en KEYWORD, TEXT*
> - *Le champ Type BOOLEAN peut être modifié en KEYWORD, TEXT*
> - *Le champ Type GEO-POINT peut être modifié en KEYWORD, TEXT*
> - *Le champ Type ENUM de valeur peut être modifié en KEYWORD, TEXT*

5.12.1.1. Sauvegarde du JSON STP_BACKUP_ONTOLOGY

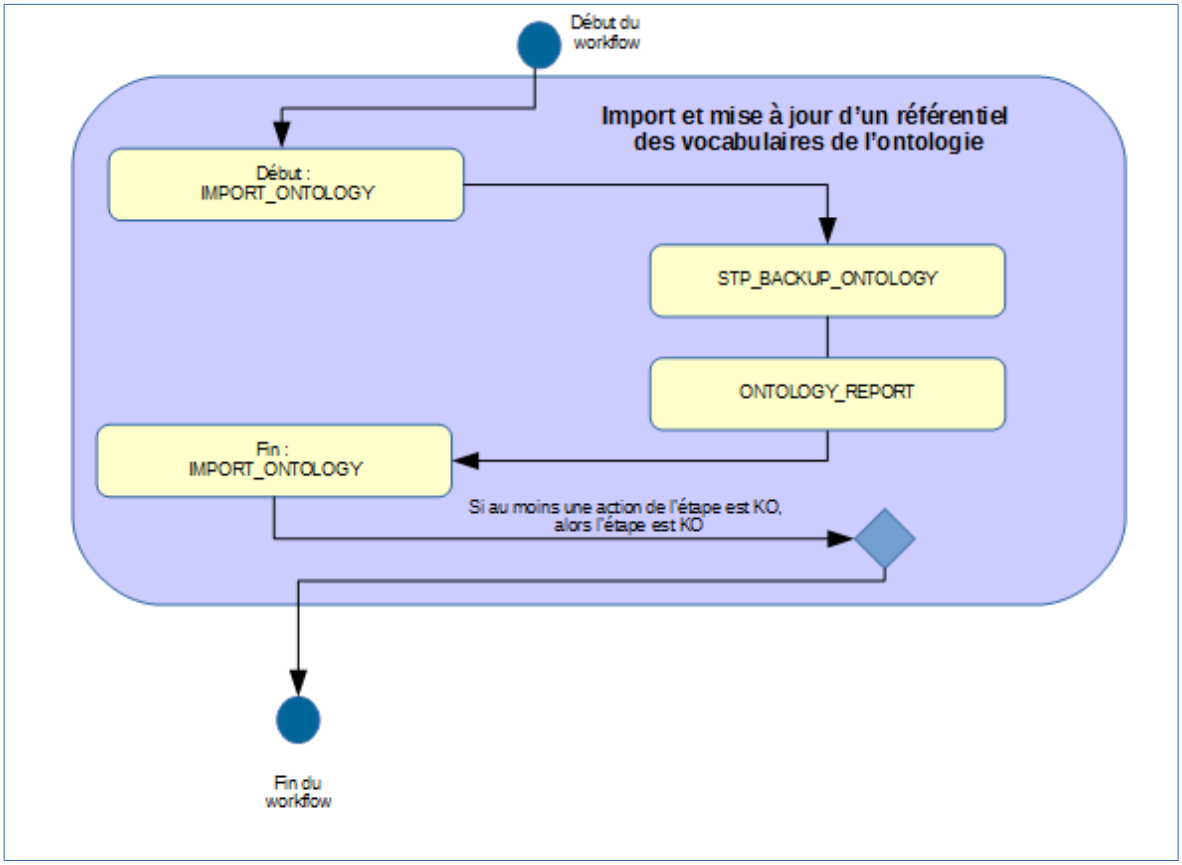
- **Règle** : tâche consistant à enregistrer une copie de la base de données des métadonnées des ontologies sur les offres de stockage
- **Type** : bloquant
- **Statuts** :
  - OK : une copie de la base de données nouvellement importée est enregistrée (STP_BACKUP_ONTOLOGY.OK = Succès du processus de sauvegarde du référentiel des ontologies)
  - KO : pas de cas KO
  - FATAL : une erreur technique est survenue lors de la copie de la base de données (STP_BACKUP_ONTOLOGY.FATAL = Erreur technique lors du processus de sauvegarde du référentiel des ontologies)

5.12.1.2. Processus de génération du rapport d'import du référentiel des vocabulaires de l'ontologie ONTOLOGY_REPORT

- **Règle** : tâche consistant à créer le rapport d'import du référentiel des vocabulaires de l'ontologie
- **Type** : bloquant
- **Statuts** :
  - OK : le rapport d'import du référentiel des vocabulaires de l'ontologie a bien été créé (ONTOLOGY_REPORT.OK = Succès du processus de génération du rapport d'import du référentiel des vocabulaires de l'ontologie)
  - KO : pas de KO
  - FATAL : une erreur technique est survenue lors de la création du rapport d'import du référentiel des vocabulaires de l'ontologie (ONTOLOGY_REPORT .FATAL = Erreur technique lors du processus de génération du rapport d'import du référentiel des vocabulaires de l'ontologie)

5.12.2. Structure du Workflow d'import et de mise à jour d'un référentiel des vocabulaires de l'ontologie

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



5.12.3. Structure du rapport d'administration des vocabulaires de l'ontologie

Lorsqu'un référentiel est importé ou mis à jour, la solution logicielle Vitam génère un rapport de l'opération.

Ce rapport est en plusieurs parties :

- « Operation » contient :
  - « evType » : le type d'opération. Dans le cadre de ce rapport, il s'agit toujours de « IMPORT_ONTOLOGY »

- « evDateTime » : la date et l'heure de l'opération d'import
- « evId » : l'identifiant de l'opération
- « outMessage » : le type d'opération. Dans le cadre de ce rapport, il s'agit toujours de « IMPORT_ONTOLOGY »

- « deteledOntologies » : liste des vocabulaires supprimés
- « updatedOntologies » : liste des vocabulaires mis à jour
- « createdOntologies » : liste des vocabulaires ajoutés

Exemple :


```
{
  "Operation": {
     "evType": "IMPORT_ONTOLOGY",
     "evDateTime": "2019-11-09T18:54:23.097",
     "evId": "aeeaaaaaaghia7juabddealokgb75gyaaaaq",
     "outMessg": "IMPORT_ONTOLOGY"
  },
  "deletedOntologies": [],
  "updatedOntologies": ["AcquiredDate", "BirthDate", "BirthName", "Address", "City",
"Country", "Geogname", "PostalCode", "Region", "Corpname", "DeathDate", "FirstName",
"Gender", "GivenName", "Identifier", "Nationality",
"ArchivalAgencyArchiveUnitIdentifier", "ArchiveUnitProfile", "Juridictional", "Spatial",
"Temporal", "CreatedDate", "DataObjectGroupReferenceId", "CustodialHistoryItem",
"Description", "DescriptionLanguage", "DescriptionLevel", "DocumentType", "EndDate",
"evTypeDetail", "FilePlanPosition", "GpsAltitude", "GpsAltitudeRef", "GpsDateStamp",
"GpsLatitude", "GpsLatitudeRef", "GpsLongitude", "GpsLongitudeRef", "GpsVersionID",
"KeywordContent", "KeywordReference", "KeywordType", "Language",
"OriginatingAgencyArchiveUnitIdentifier", "OriginatingSystemId", "ReceivedDate",
"RegisteredDate", "ArchiveUnitRefId", "DataObjectReferenceId",
```

```
"RepositoryArchiveUnitPID", "RepositoryObjectPID", "ExternalReference", "Activity",
"ExecutableName", "ExecutableVersion", "Function", "Position", "Role", "Mandate",
"SentDate", "Algorithm", "SignedObjectId", "FullName", "SigningTime", "ValidationTime",
"Source", "StartDate", "Status", "SystemId", "Tag", "Title", "TransactedDate",
"TransferringAgencyArchiveUnitIdentifier", "Type", "Version", "_glpd", "_graph", "_max",
"PreventInheritance", "PreventRulesId", "Rule", "FinalAction", "ClassificationLevel",
"ClassificationOwner", "ClassificationAudience", "ClassificationReassessingDate",
"NeedReassessingAuthorization", "NeedAuthorization", "_min", "_nbc", "_og", "_opi",
"_ops", "_opts", "_sp", "_sps", "offerIds", "strategyId", "_tenant", "_unitType", "_up",
"_us", "_v", "_av", "CreatingApplicationName", "CreatingApplicationVersion",
"CreatingOs", "CreatingOsVersion", "DateCreatedByApplication", "Filename",
"LastModified", "_profil", "qualifier", "DataObjectGroupId", "DataObjectVersion",
"Encoding", "FormatId", "FormatLitteral", "MimeType", "MessageDigest", "dValue", "unit",
"NumberOfPage", "Shape", "PhysicalId", "Size", "Uri", "_id", "AccessLog",
"ActivationDate", "CreationDate", "DeactivationDate", "EveryDataObjectVersion",
"EveryOriginatingAgency", "ExcludedRootUnits", "LastUpdate", "Name",
"OriginatingAgencies", "RuleCategoryToFilter", "RootUnits", "WritingPermission",
"WritingRestrictedDesc", "AcquisitionInformation", "ArchivalAgreement", "deleted",
"ActionList", "FormatList", "GriffinIdentifier", "Timeout", "MaxSize", "Debug", "Args",
"ingested", "remained", "Opi", "Opc", "OpType", "Gots", "Units", "Objects", "ObjSize",
"OperationIds", "OriginatingAgency", "SubmissionAgency", "LegalStatus", "ControlSchema",
"Fields", "EnableControl", "AccessContracts", "IngestContracts", "tenant",
"SecurityProfile", "Alert", "Comment", "Extension", "Group",
"HasPriorityOverFileFormatID", "PUID", "VersionPronom", "ArchiveProfiles",
"CheckParentLink", "EveryFormatType", "FormatType", "FormatUnidentifiedAuthorized",
"ComputeInheritedRulesAtIngest", "LinkParentId", "MasterMandatory",
"ManagementContractId", "_lastPersistedDate", "agId", "evDateTime", "evId", "evDetData",
"evIdProc", "SecurisationVersion", "MaxEntriesReached", "ServiceLevel", "evIdReq",
"evParentId", "evType", "evTypeProc", "ArchivalAgency", "TransferringAgency",
"AgIfTrans", "EvDateTimeReq", "EvDetailReq", "Hash", "LogType",
"MinusOneMonthLogbookTraceabilityDate", "MinusOneYearLogbookTraceabilityDate",
"NumberOfElements", "PreviousLogbookTraceabilityDate", "TimeStampToken", "diff",
"errors", "reports", "loadingURI", "pointer", "obId", "outDetail", "outMessg",
"outcome", "ApiField", "Collections", "Origin", "SedaField", "ShortName", "Format",
"Path", "RuleDescription", "RuleDuration", "RuleId", "RuleMeasurement", "RuleType",
"RuleValue", "UpdateDate", "FullAccess", "Permissions", "ud", "OperationId",
"GlobalStatus", "DestroyableOriginatingAgencies", "NonDestroyableOriginatingAgencies",
"ExtendedInfoType", "ParentUnitId", "OriginatingAgenciesInConflict", "_sedaVersion",
"_implementationVersion", "Compressed", "RawMetadata", "BinaryObjectSize",
"BinaryObject", "ObjectGroup", "ArchiveUnit", "FilteredExtractedObjectGroupData",
"FilteredExtractedUnitData", "CheckParentId", "ArchivalProfile", "DataObjectSystemId",
"DataObjectGroupSystemId", "_validComputedInheritedRules", "MaxEndDate",
"indexationDate", "UnitStrategy", "ObjectGroupStrategy", "ObjectStrategy"],

  "createdOntologies": ["MyKeyword", "MyText", "MyDate", "MyBoolean", "MyLong",
"MyDouble", "MyGeoPoint", "MyEnum"]

}
```

5.13.Workflow d'administration d'un référentiel des griffons

Cette section décrit le processus permettant d'importer des griffons. Cette opération n'est réalisable que sur le tenant d'administration.

Toutes les étapes, tâches et traitements sont journalisées dans le journal des opérations et décrivent le processus (clé et description de la clé associée dans le journal des opérations) tel qu'implémenté dans la version actuelle de la solution logicielle Vitam.

5.13.1.Processus d'import et mise à jour des griffons (STP_IMPORT_GRIFFIN)

Le processus d'import d'un référentiel des griffons permet d'ajouter des griffons qui seront utilisés dans les opérations de préservation. Tous les éléments réalisés au cours de ce processus sont exécutés dans une seule étape.

- **Règle** : opération consistant à vérifier la présence des informations minimales dans le référentiel des griffons, à s'assurer de la cohérence des informations saisies, à affecter des données aux champs peuplés par la solution logicielle Vitam et à enregistrer le référentiel des griffons.

  Les données suivantes sont obligatoirement remplies :
  - Le champ « Name » est peuplé d'une chaîne de caractères ;
  - Le champ « Identifier » est peuplé d'une chaîne de caractères devant correspondre à un identifiant signifiant donné au griffon ;
  - Le champ « ExecutableName » est peuplé d'une chaîne de caractères devant correspondre au nom technique du griffon utilisé pour lancer l'exécutable dans le système ;
  - Le champ « ExecutableVersion » est peuplé d'une valeur devant correspondre à la version du griffon utilisé. Un même exécutable (ExecutableName) peut être associé à plusieurs versions.
  - Les données suivantes optionnelles si elles sont remplies le sont en respectant les règles énoncées pour chacune :
    - Le champ « Description » doit être une chaîne de caractères,
    - Le champ « CreationDate » doit être une date est au format ISO 8601,
    - Le champ « LastUpdate » doit être une date est au format ISO 8601,
    - Le champ « _tenant » doit être un entier,
    - Le champ « _v » doit être un entier.

La modification d'un référentiel des griffons s'effectue par ré-import du fichier Json. Le nouvel import annule et remplace le référentiel précédent. Ce ré-import observe les règles décrites dans le processus d'import.

5.13.1.1. Sauvegarde du JSON STP_BACKUP_GRIFFIN

- **Règle** : tâche consistant à enregistrer une copie de la base de données des métadonnées des griffons sur les offres de stockage
- **Type** : bloquant
- **Statuts** :
  - OK : une copie de la base de données nouvellement importée est enregistrée (STP_BACKUP_GRIFFIN.OK = Succès du processus de sauvegarde des griffons)
  - KO : pas de cas KO
  - FATAL : une erreur technique est survenue lors de la copie de la base de données (STP_BACKUP_GRIFFIN.FATAL = Erreur technique lors du processus de sauvegarde des griffons

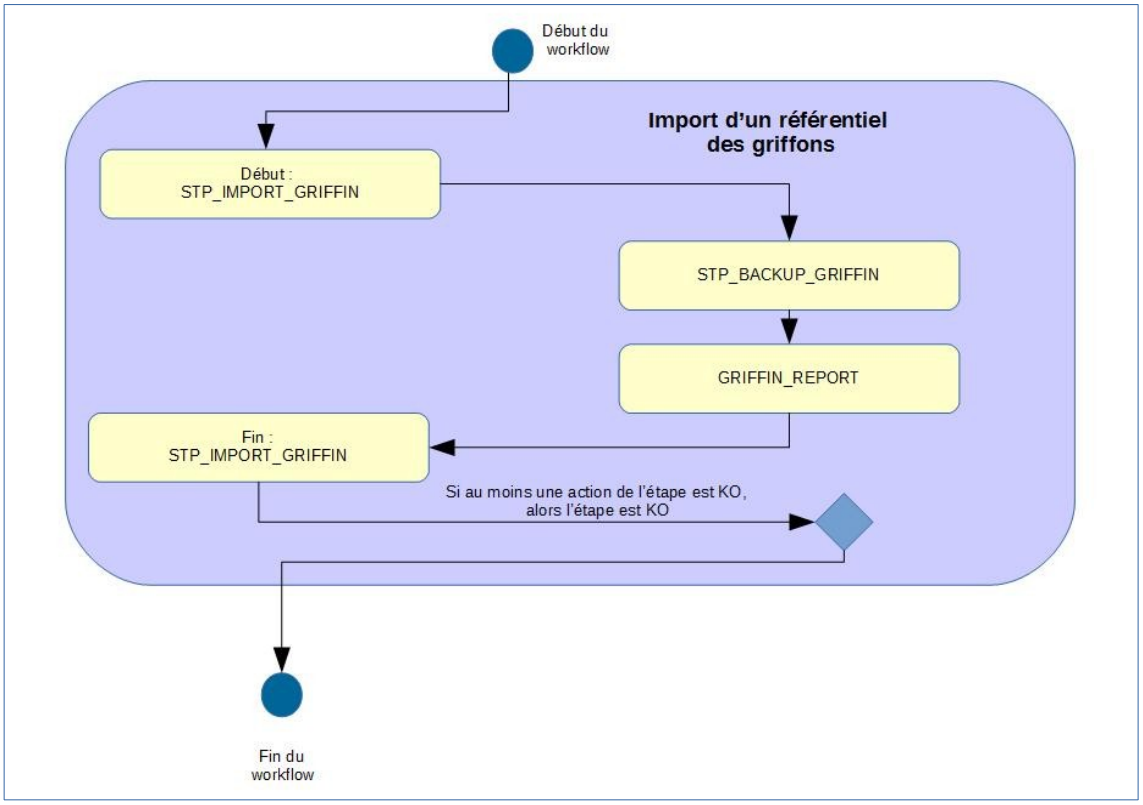
5.13.1.2. Processus de génération du rapport d'import du référentiel des griffons GRIFFIN_REPORT

- **Règle** : tâche consistant à créer le rapport d'import du référentiel des griffons

- **Type** : bloquant
- **Statuts** :
  - OK : le rapport d'import du référentiel des griffons a bien été créé (GRIFFIN_REPORT.OK = Succès du processus de génération du rapport d'import du référentiel des griffons)
  - KO : pas de KO
  - FATAL : une erreur technique est survenue lors de la création du rapport d'import du référentiel des griffons (GRIFFIN_REPORT.FATAL = Erreur technique lors du processus de génération du rapport d'import du référentiel des griffons)

5.13.2. Structure du Workflow d'import d'un référentiel des griffons

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



5.13.3. Structure du rapport d'administration du référentiel des griffons

Lorsqu'un référentiel des griffons est importé ou mis à jour, la solution logicielle Vitam génère un rapport de l'opération.

Ce rapport est en plusieurs parties :

- « Operation » contient :
  - « evType » : le type d'opération. Dans le cadre de ce rapport, il s'agit toujours de « IMPORT_GRIFFIN » ;
  - « evDateTime » : la date et l'heure de l'opération d'import ;
  - « evId » : l'identifiant de l'opération ;
- « StatusCode » : le statut de l'opération OK, KO, WARNING ;
- « PreviousGriffinsVersion » : le numéro de la version précédemment installée dans le référentiel ;

- « PreviousGriffinCreationDate » : la date de la version précédemment installée dans le référentiel ;
- « NewGriffinsVersion » : le numéro de version désormais installée ;
- « RemovedIdentifiers » : la liste des griffons qui ont été supprimés ;
- « AddedIdentifiers » : la liste des griffons qui ont été ajoutés ;
- « UpdatedIdentifiers » : la liste des griffons qui ont été mis à jour ;
- « Warnings » : les messages d'avertissement.

Exemple :


```
{  "Operation" : {"evType" : "IMPORT_GRIFFIN","evDateTime" : "2019-01-
30T12:58:05.176","evId" : "aeeaaaaaaghfj4pcabeloalit3lip3yaaaaq"},
  "StatusCode" : "WARNING",
  "PreviousGriffinsVersion" : "V1",
  "PreviousGriffinsCreationDate" : "2019-01-30T12:58:05.377",
  "NewGriffinsVersion" : "V1.0.0",
  "RemovedIdentifiers" : [ "GRIFFIN2", "GRIFFIN1", "GRI-000005", "GRIFFIN3" ],
  "AddedIdentifiers" : [ ],
  "UpdatedIdentifiers" : {"GRI-000002" : [ ],"GRI-000001" : [ ]},
  "Warnings" : [ "4 identifiers removed." ]}
```

5.14.Workflow d'administration d'un référentiel des scénarios de préservation

Cette section décrit le processus permettant d'importer un référentiel des scénarios de préservation.

Toutes les étapes, tâches et traitements sont journalisées dans le journal des opérations et décrivent le processus (clé et description de la clé associée dans le journal des opérations) tel qu'implémenté dans la version actuelle de la solution logicielle Vitam.

5.14.1.Processus d'import et de mise à jour des scénarios de préservation (STP_IMPORT_ SCENARIO)

Le processus d'import d'un référentiel de scénarios de préservation permet d'ajouter des scénarios de préservation qui seront utilisés dans les opérations de préservation. Tous les éléments réalisés au cours de ce processus sont exécutés dans une seule étape.

- **Règle** : opération consistant à vérifier la présence des informations minimales dans le référentiel des scénarios de préservation, à s'assurer de la cohérence des informations saisies, à affecter des données aux champs peuplés par la solution logicielle Vitam et à enregistrer le référentiel des scénarios de préservation.

  Les données suivantes sont obligatoirement remplies :
  - Le champ « Name » doit être peuplé avec une chaîne de caractères unique ;
  - Le champ « Identifier » doit être unique ;
  - Les champs « ActionList » et « Type » doivent avoir pour valeur « ANALYSE », « GENERATE », « IDENTIFY », « EXTRACT » ou « EXTRACT » ;
  - Le champ « FormatList » doit avoir pour valeur une suite de chaînes de caractères correspondant à une valeur valide issue du champ « PUID » du référentiel des formats ;
  - le champ « GriffinIdentifier » doit avoir pour valeur une chaîne de caractère correspondant à une valeur valide issue du champ « Identifier » du référentiel des griffons ;
  - Les champs « Timeout » et « MaxSize » doivent avoir pour valeur un entier ;
  - Le champ « Debug » doit être un booléen et avoir pour valeur « true » ou « false » ;
  - Le champ « Extension » doit avoir pour valeur une chaîne de caractères ;
  - Le champ « Args » doit avoir pour valeur une chaîne de caractères ;
  - Les données suivantes sont facultatives ; si elles sont remplies, elles respectent les règles énoncées pour chacune :
    - Le champ « Description » est peuplé d'une chaîne de caractères.

Exemple :

```
[
  {
    "Identifier": "PSC-000002",
    "Name": "Transformation en GIF MINI",
    "Description": "Ce scenario transforme une image JPEG en GIF mini",
    "CreationDate": "2018-11-16T15:55:30.721",
    "LastUpdate": "2018-11-20T15:34:21.542",
    "ActionList": [
      "GENERATE"
    ],
    "GriffinByFormat": [
      {
        "FormatList": ["fmt/41", "fmt/43"],
        "GriffinIdentifier": "GRI-000001",
        "TimeOut": 20,
        "MaxSize": 10000000,
        "Debug":true,
        "ActionDetail": [
          {
            "Type": "GENERATE",
            "Values": {
```

```
              "Extension": "GIF",
              "Args": [
                "-thumbnail",
                "100x100"
              ]
            }
          }
        ]
      }
    ],
    "DefaultGriffin": null
  }
]
```

5.14.1.1. Sauvegarde du JSON STP_BACKUP_SCENARIO

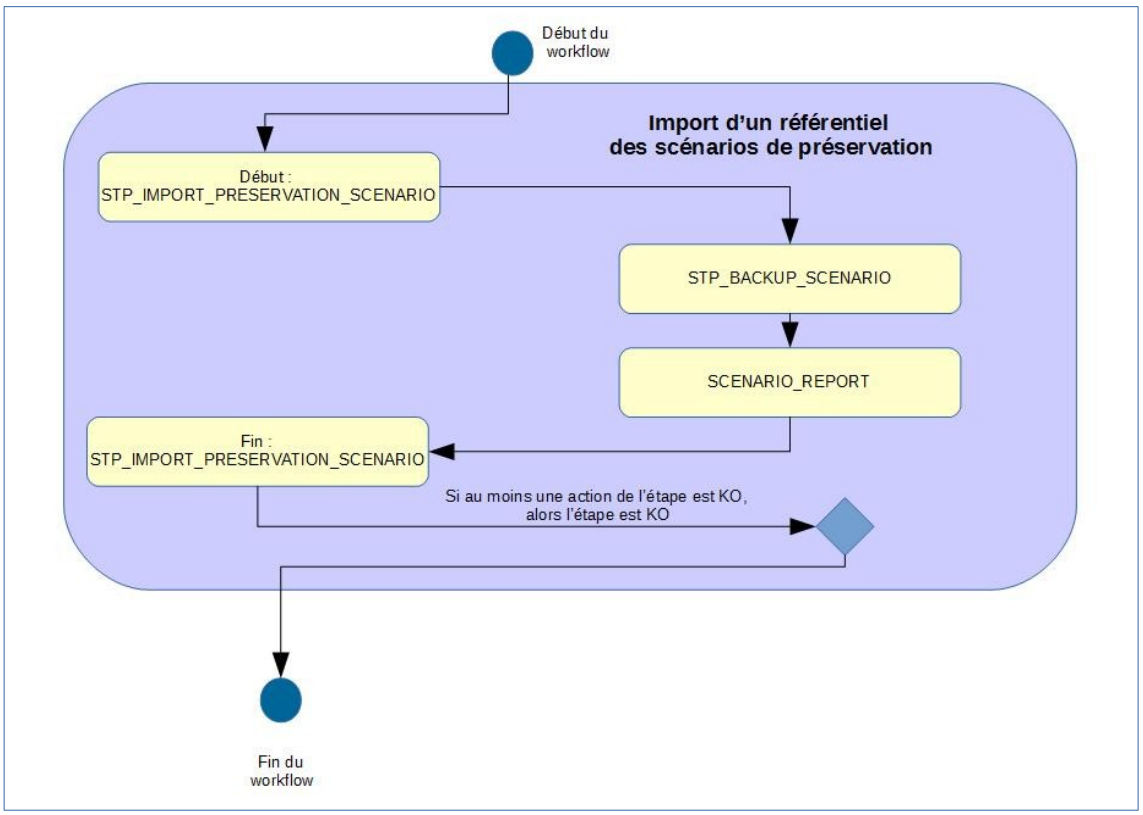
- **Règle** : tâche consistant à enregistrer une copie de la base de données des scénarios de préservation sur les offres de stockage
- **Type** : bloquant
- **Statuts** :
  - OK : une copie de la base de données nouvellement importée est enregistrée (STP_BACKUP_SCENARIO.OK = Succès du processus de sauvegarde des scénarios de préservation)
  - KO : pas de KO
  - FATAL : une erreur technique est survenue lors de la sauvegarde des scénarios de préservation (BACKUP_SCENARIO.FATAL = Erreur technique lors du processus de sauvegarde des scénarios de préservation)

5.14.1.2. Processus de génération du rapport d'import du référentiel des scénarios de préservation SCENARIO_REPORT

- **Règle** : tâche consistant à créer le rapport d'import du référentiel des scénarios de préservation
- **Type** : bloquant
- **Statuts** :
  - OK : le rapport d'import du référentiel des scénarios de préservation a bien été créé (SCENARIO_REPORT.OK = Succès du processus de génération du rapport d'import du référentiel des scénarios de préservation)
  - KO : pas de KO
  - FATAL : une erreur technique est survenue lors de la création du rapport d'import du référentiel des scénarios de préservation (SCENARIO_REPORT .FATAL = Erreur technique lors du processus de génération du rapport d'import du référentiel des scénarios de préservation)

5.14.2. Structure du Workflow d'import d'un référentiel des scénarios de préservation

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



5.14.3. Structure du rapport d'administration du référentiel des scénarios de préservation

Lorsqu'un référentiel des scénarios de préservation est importé ou mis à jour, la solution logicielle Vitam génère un rapport de l'opération.

Ce rapport est en plusieurs parties :

- « Operation » contient :
  - « evType » : le type d'opération. Dans le cadre de ce rapport, il s'agit toujours de « STP_IMPORT_PRESERVATION_SCENARIO » ;
  - « evDateTime » : la date et l'heure de l'opération d'import ;
  - « evId » : l'identifiant de l'opération ;
- « StatusCode » : le statut de l'opération OK, KO, WARNING ;
- « PreviousScenariosCreationDate » : la date de la version précédemment installée dans le référentiel ;
- « NewScenariosCreationDate » : la date de la version installée dans le référentiel ;
- « RemovedIdentifiers » : la liste des scénarios qui ont été supprimés ;
- « AddedIdentifiers » : la liste des scénarios qui ont été ajoutés ;
- « UpdatedIdentifiers » : la liste des scénarios qui ont été mis à jour ;
- « Warnings » : les messages d'avertissement.

Exemple :

```
{  "Operation" : {"evType" : "IMPORT_GRIFFIN","evDateTime" : "2019-01-
30T12:58:05.176","evId" : "aeeaaaaaaghfj4pcabeloalit3lip3yaaaaq"},
  "StatusCode" : "WARNING",
```

```
"PreviousScenariosCreationDate" : "2019-01-30T12:58:05.377",
"NewScenariosCreationDate" : "2019-01-30T12:58:05.377",
"RemovedIdentifiers" : [ "SC2", "SC1", "SCENARIO5", "SCENARIO3" ],
"AddedIdentifiers" : [ ],
"UpdatedIdentifiers" : {"SCE-000002" : [ ],"SCE-000001" : [ ]},
"Warnings" : [ "4 identifiers removed." ]}
```