VITAM - Documentation d'exploitation

Version 2.15.1-1

VITAM

déc. 11, 2019

Table des matières

CHAPITRE 1

Introduction

1.1 But de cette documentation

Ce document a pour but de permettre de fournir à une équipe d'exploitants de la solution logicielle *VITAM* les procédures et informations utiles et nécessaires au bon fonctionnement de la solution logicielle.

1.2 Destinataires de ce document

Ce document s'adresse à des exploitants du secteur informatique ayant de bonnes connaissances en environnement Linux.

CHAPITRE 2

Rappels

2.1 Information concernant les licences

La solution logicielle *VITAM* est publiée sous la licence CeCILL 2.1 [1] ; la documentation associée (comprenant le présent document) est publiée sous Licence Ouverte V2.0 [2].

2.2 Documents de référence

2.2.1 Documents internes

Tableau 1 – Documents de référence VITAM

Nom	Lien
DAT	http://www.programmevitam.fr/ressources/DocCourante/html/archi
DIN	http://www.programmevitam.fr/ressources/DocCourante/html/installation
DEX	http://www.programmevitam.fr/ressources/DocCourante/html/exploitation
DMV	http://www.programmevitam.fr/ressources/DocCourante/html/migration
Release notes	https://github.com/ProgrammeVitam/vitam/releases/latest

2.2.2 Référentiels externes

2.3 Glossaire

API *Application Programming Interface*

AU *Archive Unit*, unité archivistique

[1] http://www.cecill.info/licences/Licence_CeCILL_V2.1-fr.html
[2] https://www.etalab.gouv.fr/wp-content/uploads/2017/04/ETALAB-Licence-Ouverte-v2.0.pdf

BDD Base De Données

BDO *Binary DataObject*

CA *Certificate Authority*, autorité de certification

CAS Content Adressable Storage

CCFN Composant Coffre Fort Numérique

CN Common Name

COTS Component Off The shelf ; il s'agit d'un composant « sur étagère », non développé par le projet *VITAM*, mais intégré à partir d'un binaire externe. Par exemple : MongoDB, ElasticSearch.

CRL *Certificate Revocation List* ; liste des identifiants des certificats qui ont été révoqués ou invalidés et qui ne sont donc plus dignes de confiance. Cette norme est spécifiée dans les RFC 5280 et RFC 6818.

CRUD *create, read, update, and delete*, s'applique aux opérations dans une base de données MongoDB

DAT Dossier d'Architecture Technique

DC Data Center

DEX Dossier d'EXploitation

DIN Dossier d'INstallation

DIP *Dissemination Information Package*

DMV Documentation de Montées de Version

DNS *Domain Name System*

DNSSEC *Domain Name System Security Extensions* est un protocole standardisé par l'IETF permettant de résoudre certains problèmes de sécurité liés au protocole DNS. Les spécifications sont publiées dans la RFC 4033 et les suivantes (une version antérieure de DNSSEC n'a eu aucun succès). Définition DNSSEC [3]

DSL *Domain Specific Language*, language dédié pour le requêtage de VITAM

DUA Durée d'Utilité Administrative

EBIOS Méthode d'évaluation des risques en informatique, permettant d'apprécier les risques Sécurité des systèmes d'information (entités et vulnérabilités, méthodes d'attaques et éléments menaçants, éléments essentiels et besoins de sécurité...), de contribuer à leur traitement en spécifiant les exigences de sécurité à mettre en place, de préparer l'ensemble du dossier de sécurité nécessaire à l'acceptation des risques et de fournir les éléments utiles à la communication relative aux risques. Elle est compatible avec les normes ISO 13335 (GMITS), ISO 15408 (critères communs) et ISO 17799

EAD Description archivistique encodée

ELK *Elasticsearch Logstash Kibana*

FIP *Floating IP*

GOT Groupe d'Objet Technique

IHM Interface Homme Machine

IP *Internet Protocol*

IsaDG Norme générale et internationale de description archivistique

JRE *Java Runtime Environment* ; il s'agit de la machine virtuelle Java permettant d'y exécuter les programmes compilés pour.

JVM *Java Virtual Machine* ; Cf. *JRE*

LAN *Local Area Network*, réseau informatique local, qui relie des ordinateurs dans une zone limitée

LFC *LiFe Cycle*, cycle de vie

LTS *Long-term support*, support à long terme : version spécifique d'un logiciel dont le support est assuré pour une période de temps plus longue que la normale.

M2M *Machine To Machine*

[3] https://fr.wikipedia.org/wiki/Domain_Name_System_Security_Extensions

MitM L'attaque de l'homme du milieu (HDM) ou *man-in-the-middle attack* (MITM) est une attaque qui a pour but d'intercepter les communications entre deux parties, sans que ni l'une ni l'autre ne puisse se douter que le canal de communication entre elles a été compromis. Le canal le plus courant est une connexion à Internet de l'internaute lambda. L'attaquant doit d'abord être capable d'observer et d'intercepter les messages d'une victime à l'autre. L'attaque « homme du milieu » est particulièrement applicable dans la méthode d'échange de clés Diffie-Hellman, quand cet échange est utilisé sans authentification. Avec authentification, Diffie-Hellman est en revanche invulnérable aux écoutes du canal, et est d'ailleurs conçu pour cela. Explication [4]

MoReq *Modular Requirements for Records System*, recueil d'exigences pour l'organisation de l'archivage, élaboré dans le cadre de l'Union européenne.

NoSQL Base de données non-basée sur un paradigme classique des bases relationnelles. Définition NoSQL [5]

NTP *Network Time Protocol*

OAIS *Open Archival Information System*, acronyme anglais pour Systèmes de transfert des informations et données spatiales – Système ouvert d'archivage d'information (SOAI) - Modèle de référence.

OOM Aussi apelé *Out-Of-Memory Killer* ; mécanisme de la dernière chance incorporé au noyau Linux, en cas de dépassement de la capacité mémoire

OS *Operating System*, système d'exploitation

OWASP *Open Web Application Security Project*, communauté en ligne de façon libre et ouverte à tous publiant des recommandations de sécurisation Web et de proposant aux internautes, administrateurs et entreprises des méthodes et outils de référence permettant de contrôler le niveau de sécurisation de ses applications Web

PDMA Perte de Données Maximale Admissible ; il s'agit du pourcentage de données stockées dans le système qu'il est acceptable de perdre lors d'un incident de production.

PKI Une infrastructure à clés publiques (ICP) ou infrastructure de gestion de clés (IGC) ou encore Public Key Infrastructure (PKI), est un ensemble de composants physiques (des ordinateurs, des équipements cryptographiques logiciels ou matériel type HSM ou encore des cartes à puces), de procédures humaines (vérifications, validation) et de logiciels (système et application) en vue de gérer le cycle de vie des certificats numériques ou certificats électroniques. Définition PKI [6]

PCA Plan de Continuité d'Activité

PRA Plan de Reprise d'Activité

REST *REpresentational State Transfer* : type d'architecture d'échanges. Appliqué aux services web, en se basant sur les appels http standard, il permet de fournir des API dites « RESTful » qui présentent un certain nombre d'avantages en termes d'indépendance, d'universalité, de maintenabilité et de gestion de charge. Définition REST [7]

RGAA Référentiel Général d'Accessibilité pour les Administrations

RGI Référentiel Général d'Interopérabilité

RPM *Red Hat Package Manager* ; il s'agit du format de packets logiciels nativement utilisé par les distributions CentOS (entre autres)

SAE Système d'Archivage Électronique

SEDA Standard d'Échange de Données pour l'Archivage

SGBD Système de Gestion de Base de Données

SGBDR Système de Gestion de Base de Données Relationnelle

SIA Système d'Informations Archivistique

SIEM *Security Information and Event Management*

SIP *Submission Information Package*

SSH *Secure SHell*

[4] https://fr.wikipedia.org/wiki/Attaque_de_l'homme_du_milieu
[5] https://fr.wikipedia.org/wiki/NoSQL
[6] https://fr.wikipedia.org/wiki/Infrastructure_%C3%A0_cl%C3%A9s_publiques
[7] https://fr.wikipedia.org/wiki/Representational_state_transfer

Swift *OpenStack Object Store project*

TLS *Transport Layer Security*

TNR Tests de Non-Régression

TTL *Time To Live*, indique le temps pendant lequel une information doit être conservée, ou le temps pendant lequel une information doit être gardée en cache

UDP *User Datagram Protocol*, protocole de datagramme utilisateur, un des principaux protocoles de télécommunication utilisés par Internet. Il fait partie de la couche transport du modèle OSI

UID *User IDentification*

VITAM Valeurs Immatérielles Transférées aux Archives pour Mémoire

VM *Virtual Machine*

WAF *Web Application Firewall*

WAN *Wide Area Network*, réseau informatique couvrant une grande zone géographique, typiquement à l'échelle d'un pays, d'un continent, ou de la planète entière

CHAPITRE 3

Expertises requises

Les équipes en charge du déploiement et de l'exploitation de la solution logicielle *VITAM* devront disposer en interne des compétences suivantes :

- connaissance d'ansible en tant qu'outil de déploiement automatisé ;
- connaissance de Consul en tant qu'outil de découverte de services ;
- maîtrise de MongoDB et ElasticSearch par les administrateurs de bases de données.

CHAPITRE 4

Architecture de la solution logicielle VITAM

Le schéma ci-dessous représente une solution logicielle *VITAM* :

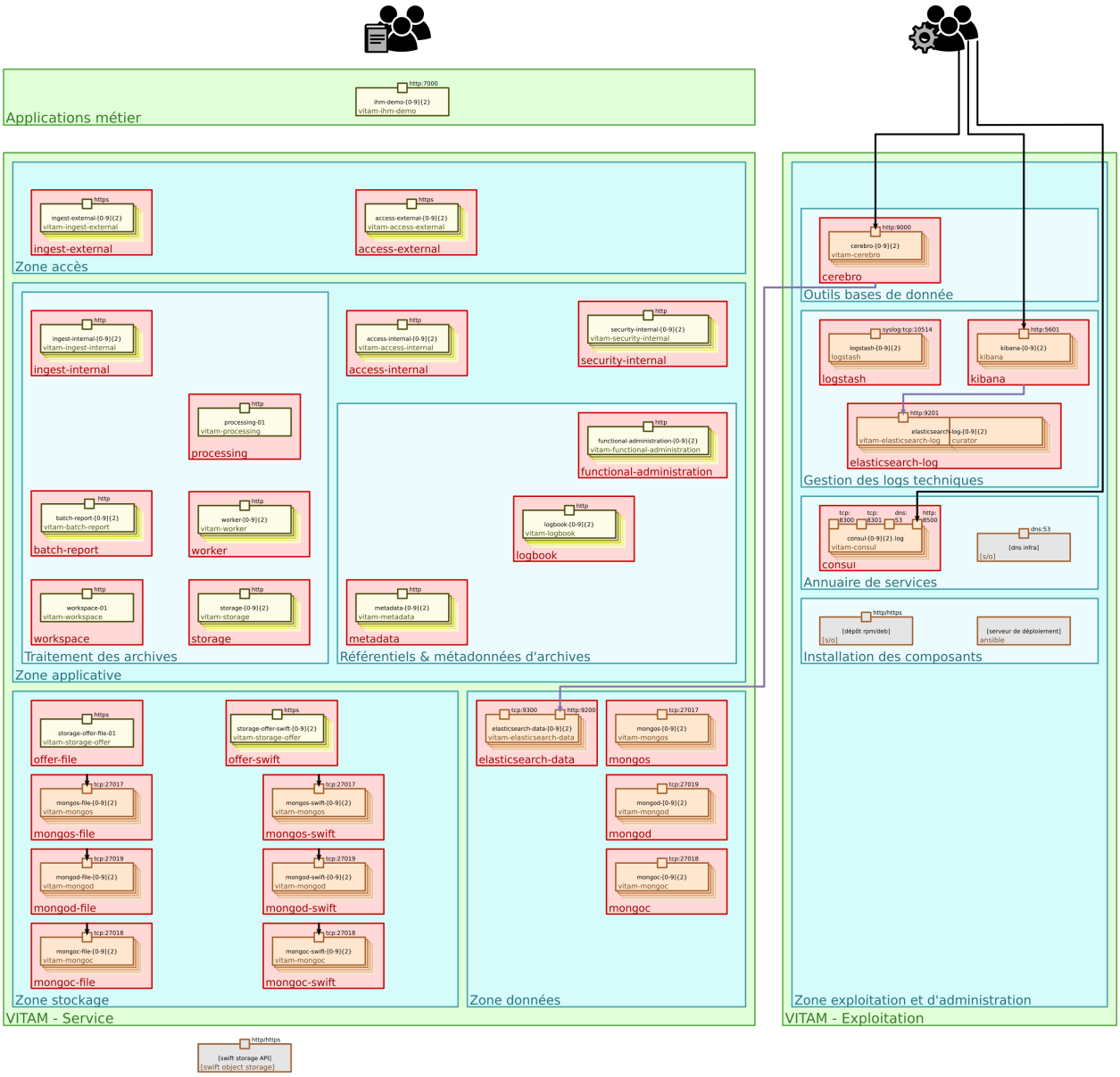


Fig. 1 – Vue d'ensemble d'un déploiement *VITAM* : zones, composants

Voir aussi :

Se référer au *DAT* (et notamment le chapitre dédié à l'architecture technique) pour plus de détails, en particulier concernant les flux entre les composants.

CHAPITRE 5

Exploitation globale

5.1 Gestion des accès

5.1.1 API

La gestion des accès à aux *API* externes se fait via les *granted stores* (cf. *Fichiers communs* (page 100) pour les fichiers communs, cf. *Configuration / fichiers utiles* (page 128) pour access-external et *Configuration / fichiers utiles* (page 157) pour ingest-external).

5.1.2 IHM de démonstration

Dans cette version, la gestion et l'authentification des utilisateurs peut se faire par :

- un fichier plat qui contient logins et mots de passe (peut sécurisé)
- certificat x509
- un ldap

5.1.3 IHM de recette

Dans cette version, la gestion des utilisateurs se fait par :

- un fichier plat qui contient logins et mots de passe (peu sécurisé)
- certificat x509
- un ldap

5.2 Portails d'administration

5.2.1 Technique

Aucun portail d'administration technique n'est prévu dans cette version de la solution logicielle *VITAM*.

5.2.2 Fonctionnel

Le portail d'administration fonctionnel est intégré au composant **ihm-demo** dans cette version de la solution logicielle *VITAM* (cf. *Présentation* (page 141)).

5.3 Paramétrage & configuration

L'étape de paramétrage et la configuration sont essentiellement liées à la mise en place ou la mise à niveau de la solution logicielle *VITAM* (ansible / inventaire).

Voir aussi :

Plus d'informations, et notamment les paramètres d'installation, sont disponibles dans le *DIN*.

5.3.1 Mise à niveau de la configuration de l'environnement

5.3.1.1 Mise à jour du nombre de tenants

Note : se référer au *DIN*

5.3.1.2 Mise à jour des paramètres JVM

Note : se référer au *DIN*

5.4 Déploiement / mises à jour

5.4.1 Mise à jour des certificats

Pour mettre à jour les certificats (avant expiration par exemple), il suffit de les mettre à jour dans les répertoires de déploiement, puis de régénerer les stores (dans `environments/keystores`) et lancer leur redéploiement via cette commande ansible :

Cas où le fichier de mot de passe pour vault n'existe pas

```
# Si le mot de passe du vault n'est pas renseigné dans le fichier vault_pass.txt
ansible-playbook ansible-vitam/vitam.yml -i environments/<fichier d'inventaire> --ask-
↪vault-pass --tags update_vitam_certificates
ansible-playbook ansible-vitam-extra/extra.yml -i environments/<fichier d'inventaire>␣
↪--ask-vault-pass --tags update_vitam_certificates
```

Cas où le fichier de mot de passe pour vault existe

```
# Si le mot de passe du vault est renseigné dans le fichier vault_pass.txt
ansible-playbook ansible-vitam/vitam.yml -i environments/<fichier d'inventaire> --
↪vault-password-file vault_pass.txt --tags update_vitam_certificates
ansible-playbook ansible-vitam-extra/extra.yml -i environments/<fichier d'inventaire>␣
↪--vault-password-file vault_pass.txt --tags update_vitam_certificates
```

Voir aussi :

Le cycle de vie des certificats est rappelé dans les annexes. Une vue d'ensemble est également présentée dans le *DIN*.

5.4.2 Mise à jour de la solution logicielle VITAM

Pour la mise à jour de la solution logicielle *VITAM* (tout comme pour sa première installation), se référer au *DIN*, au *DMV*, ainsi qu'à la *release note* associée à toute version.

Ces documents détaillent les pré-requis, la configuration des fichiers et les procédures éventuelles de migration de données pour effectuer une mise à jour applicative. Le *DMV* explique également comment valider une montée de version applicative de la solution logicielle *VITAM*.

Voir aussi :

Plus d'informations, et notamment les paramètres d'installation, sont disponibles dans le *DIN*.

Voir aussi :

Dans le cadre d'une montée de version, se référer également au *DMV*.

5.4.3 Ajouter un/des instances de composants VITAM

Dans le cas où le dimensionnement initial ne donne pas pleinement satisfaction, il est possible de rajouter à une solution logicielle *VITAM* existante une/des instances supplémentaires de composants.

Pour le moment, il n'est pas possible de déplacer un composant automatiquement via ansible d'un serveur à un autre (implique une suppression du composant sur l'ancien serveur non gérée pour le moment)

Prudence : Dans le cas d'ajout d'une offre, il est nécessaire de suivre la procédure de resynchronisation des offres.

Avertissement : Seul le composant « vitam-processing » n'est pas multi-instanciable.

Avertissement : Le composant « vitam-workspace » est multi-instanciable, si son répertoire `/vitam/data/workspace` est partagé entre les différentes instances du composant. Bien penser, dans ce cas, à la problématique de droits d'accès aux fichiers et surtout aux performances de lecture/écriture sur ce disque.

1. Modifier l'inventaire avec la/les VM supplémentaire(s)
2. Lancer un déploiement comme indiqué dans le *DIN* en rajoutant la directive `-l <liste de/des VM(s) supplémentaire(s)>`

5.4.4 Modifier la fréquence de lancement de certains *timers* systemD

Par défaut, la solution logicielle *VITAM* déploie et active, selon l'usage (site primaire / site secondaire), des *timers* systemD. Le *playbook* ansible d'installation de vitam `ansible-vitam/vitam.yml`, permet d'uniquement modifier la fréquence des *timers* en rajoutant le tag `update_timers_frequency`.

Pour cela, il faut éditer la section `vitam_timers` dans le fichier `environments/group_vars/all/vitam_vars.yml`.

A l'issue, lancer le *playbook* avec la commande

```
ansible-playbook -i <inventaire> ansible-vitam/vitam.yml --tags update_timers_
↪frequency --ask-vault-pass
```

ou bien, si vous utilisez le fichier `vault_pass.txt`

```
ansible-playbook -i <inventaire> ansible-vitam/vitam.yml --tags update_timers_
↪frequency --vault-password-file vault_pass.txt
```

5.5 Interruption / maintenance

5.5.1 Procédure d'arrêt complet

Un *playbook* ansible d'arrêt complet de la solution logicielle *VITAM* est fourni, sous `deployment/ansible-vitam-exploitation` (fichier de *playbook* `stop_vitam.yml`), pour réaliser de façon automatisée les actions nécessaires.

Avertissement : Ce script, en l'état, permet un *EMERGENCY BREAK*, autrement dit un arrêt brutal des composants, ne permettant pas de garantir, à l'issue, une cohérence des données. Il est donc fortement recommandé de positionner les traitements courants en pause avant de lancer la procédure d'arrêt.

Note : Une confirmation est demandée pour lancer ce script d'arrêt de la solution logicielle VITAM.

Un playbook ansible d'arrêt des *timers* systemD *VITAM* est également fourni, sous `deployment/ansible-vitam-exploitation` (fichier de *playbook* `stop_vitam_timers.yml`), pour réaliser de façon automatisée les actions nécessaires. Ce script est à lancer une fois l'arrêt des services correctement réalisé.

5.5.2 Procédure de démarrage complet

Le pré-requis est le bon fonctionnement des partitions hébergeant la solution logicielle *VITAM*.

Un *playbook* ansible de démarrage de la solution logicielle *VITAM* est fourni, sous `deployment/ansible-vitam-exploitation` (fichier de *playbook* `start_vitam.yml`), pour réaliser de façon automatisée les actions nécessaires.

Un playbook ansible de démarrage des *timers* systemD *VITAM* est également fourni, sous `deployment/ansible-vitam-exploitation` (fichier de *playbook* `start_vitam_timers.yml`), pour réaliser de façon automatisée les actions nécessaires. Ce script est à lancer une fois le démarrage des services correctement réalisé.

5.5.3 Procédure de statut

Un playbook ansible de démarrage de VITAM est fourni, sous `deployment/ansible-vitam-exploitation` (fichier de *playbook* `status_vitam.yml`), pour réaliser de façon automatisée les tests « autotest » intégérés dans la solution logicielle *VITAM*.

5.5.4 Autres cas

5.5.4.1 Procédure de maintenance / indisponibilité de VITAM

Deux *playbooks* sont fournis dans `deployment/ansible-vitam-exploitation` :

- fichier de *playbook* `stop_external.yml`, permettant d'arrêter sélectivement les composants *VITAM* **ingest-external** et **access-external**
- fichier de *playbook* `start_external.yml`, permettant de démarrer sélectivement *VITAM* **ingest-external** et **access-external**

Ces scripts permettent d'empêcher l'usage de la solution logicielle *VITAM* par les services versants, tout en laissant opérationnel le reste de la solution logicielle. Ces *playbooks* peuvent être utiles, voire nécessaires, dans le cadre d'une migration de données ou de maintenance de la solution logicielle *VITAM*.

Ils ne stoppent par contre pas :

- Les versements qui sont encore en cours de traitement (il est toutefois possible de les mettre en pause via ihm-demo par exemple)
- Les timers qui lancent divers traitements comme des sécurisations, pour cela, se référer au chapitre suivant

5.5.4.2 Procédure de maintenance liée aux *timers systemD*

Deux *playbooks* sont fournis dans `deployment/ansible-vitam-exploitation` :

- fichier de *playbook* `stop_vitam_timers.yml`, permettant d'arrêter sélectivement les *timers systemD*
- fichier de *playbook* `start_vitam_timers.yml`, permettant de démarrer sélectivement les *timers systemD*

5.5.4.3 Procédure de maintenance sur les composants d'administration

Deux *playbooks* sont fournis dans `deployment/ansible-vitam-exploitation` :

- fichier de *playbook* `stop_vitam_admin.yml`, permettent d'arrêter sélectivement les composants Consul, la chaine de log (logstash / cluster elasticsearch log / kibana-log), cerebro et les docker mongo-express et elasticsearch-head
- fichier de *playbook* `start_vitam_admin.yml`, permettent de démarrer sélectivement les composants Consul, la chaine de log (logstash / cluster elasticsearch log / kibana-log), cerebro et les docker mongo-express et elasticsearch-head

Avertissement : En passant le *playbook* d'arrêt, l'ensemble de la solution logicielle *VITAM* devient inutilisable.

5.5.4.4 Procédure de maintenance des *IHM*

Deux *playbooks* sont fournis dans `deployment/ansible-vitam-exploitation` :

- fichier de *playbook* `stop_vitam_ihm.yml`, permettent d'arrêter sélectivement les composants *VITAM IHM* ihm-demo et ihm-recette
- fichier de *playbook* `start_vitam_ihm.yml`, permettent de démarrer sélectivement les composants *VITAM IHM* ihm-demo et ihm-recette

5.5.4.5 Procédure de maintenance des *Bases de données métier*

Quatre *playbooks* sont fournis dans `deployment/ansible-vitam-exploitation` :

- fichier de *playbook* `start_elasticsearch_data.yml`, permettent de démarrer la totalité des elasticsearch-data
- fichier de *playbook* `start_mongodb.yml`, permettent de démarrer les composants mongodb
- fichier de *playbook* `stop_elasticsearch_data.yml`, permettent d'arrêter la totalité des composants elasticsearch-data
- fichier de *playbook* `stop_mongodb.yml`, permettent d'arrêter les composants mongodb

5.6 Sauvegarde / restauration

Avertissement : Cette méthode s'applique uniquement pour des déploiements de petite taille, et n'est pas recommandée pour un usage en production. La gestion de la sauvegarde des bases de métadonnées MongoDB et Elasticsearch, ainsi que de la restauration de leur contenu en cohérence avec les offres sous-jacentes, est déjà gérée par les mécanismes intrinsèques à la solution *VITAM*, cf. le *DAT*.

Les procédures sont issues des documentations officielles :

- mongoDB : https://docs.mongodb.com/manual/tutorial/backup-and-restore-tools/
- elasticsearch : https://www.elastic.co/guide/en/elasticsearch/reference/current/modules-snapshots.html

5.6.1 Sauvegarde

Note : Pour que cette sauvegarde soit fonctionnellement correcte, il faut que la solution logicielle *VITAM* soit dans un état stable et cohérent, sans possibilité d'ajouter des « ingests » et sans travail de fond (jobs de sécurisation, . . .).

5.6.1.1 mongoDB

La commande suivante est à lancer depuis une machine hébergeant le composant `vitam-mongod` (pour data ET pour chaque offer) de *VITAM* ayant un espace suffisant dans le répertoire de sauvegarde (défini par `--out`)

```
mongodump --host mongodb.example.net --port 27017 --out /data/backup/ --username␣
↪vitamdb-admin --password "pass"
```

Note : Se reporter aux fichiers associés à l'ansiblerie de déploiement pour le mot de passe `vitamdb-admin`

Rapatrier sur un serveur approprié le produit généré dans la valeur de `--out`.

Pour rappel, il y a un cluster mongo de « data », ainsi qu'un cluster mongo « offer » associé à chaque offre. Pour un système cohérent, il faut effectuer la sauvegarde de chacun de ces *clusters*.

Note : Il est recommandé de procéder à une sauvegarde régulière des bases **identity**, **config** et **admin**.

Note : Il est recommandé de procéder à une sauvegarde régulière de la collection **identity**, ou suite à des modifications sur les certificats (ajout / mise à jour / révocation).

Un *playbook* de sauvegarde de la base de données mongoDB **identity** est fourni et stocke, sur la machine de déploiement, sous `environments/backup/identity/<date au format YYY-MM-DD>`, le produit de la commande `mongodump`.

Pour lancer le playbook

```
ansible-playbook -i environments/hosts.local ansible-vitam-exploitation/backup_
↪database_certificates.yml --vault-password-file vault_pass.txt
```

Ou si le fichier `vault_pass.txt` n'existe pas

```
ansible-playbook -i environments/hosts.local ansible-vitam-exploitation/backup_
↪database_certificates.yml --ask-vault-pass
```

5.6.1.2 Elasticsearch

La commande suivante est à lancer depuis une machine elasticsearch de *VITAM* ayant un espace suffisant dans le répertoire de sauvegarde

```
curl -X PUT http://eltasicsearch-data.service.${consul_domain}:9200/_snapshot/vitam_
↪backup -d '{ "type": "fs", "settings": { "location": "${output_dir}" } }'
```

Cette étape va créer le *repository* **vitam_backup** de sauvegarde, l'arborescence étant définie par `${output_dir}`.

Pour vérifier l'état du *repository* **vitam_backup** sur les noeuds du cluster

```
curl -X POST http://eltasicsearch-data.service.${consul_domain}:9200/_snapshot/vitam_
↪backup/_verify
```

Pour lancer un *snapshot* (dans l'exemple, appelé snapshot_1)

```
curl -X PUT http://eltasicsearch-data.service.${consul_domain}:9200/_snapshot/vitam_
↪backup/snapshot_1?wait_for_completion=true
```

Note : la commande ne rendra la main qu'à la fin de la procédure de *snapshot*.

A l'issue de la sauvegarde, procéder à une recopie de `${output_dir}` sur un serveur à part.

5.6.2 Restauration

Note : Comme pour la sauvegarde, la restauration ne peut s'effectuer que sur un environnement *VITAM* stable et cohérent, sans possibilité d'ajouter des « ingests » et sans travail de fond (jobs de sécurisation, ...). De plus, le contenu restauré doit être cohérent avec le contenu des offres de stockage sous-jacentes.

5.6.2.1 mongoDB

Il faut d'abord procéder au rapatriement dans `${output_dir}` de la sauvegarde à appliquer.

Avertissement : une sauvegarde ne peut se restaurer que sur un environnement dans la même version.

La commande suivante est à lancer depuis une machine mongo de *VITAM* possédant le répertoire de sauvegarde à restaurer :

mongorestore –host mongodb1.example.net –port 3017 –username vitamdb-admin –password "pass" ${output_dir}/${fichier}

Note : Se reporter aux fichiers associés à l'ansiblerie de déploiement pour le mot de passe `vitamdb-admin`

5.6.2.2 Elasticsearch

Il faut d'abord procéder au rapatriement dans `${output_dir}` de la sauvegarde à appliquer.

Commande pour lister les *snapshots* de **vitam_backup** (repository)

```
curl -X GET http://eltasicsearch-data.service.${consul_domain}:9200/_snapshot/vitam_
↪backup/
```

Pour lancer une restauration, placer le nom du *snapshot* sà la place de *snapshot* dans l'URL suivante

```
curl -X POST http://eltasicsearch-data.service.${consul_domain}:9200/_snapshot/vitam_
↪backup/*snapshot*/_restore
```

5.6.3 Cas de la base mongo certificates

La solution logicielle *VITAM* fournit un *playbook* de sauvegarde de la base de données `identity` ; le *backup* réalisé est stocké sur la machine de déploiement.

Pour lancer le *playbook* de sauvegarde

```
ansible-playbook -i environments/<inventaire> ansible-playbook-exploitation/backup_
↪database_certificates.yml --ask-vault-pass
```

5.7 Sauvegarde et restauration de mongodb gros volumes

5.7.1 Préconisation

Il est fortement conseillé de faire une sauvegarde (*backup*) avant toute migration (montée de version *VITAM*)

5.7.2 Sauvegarde MongoC et MongoD

5.7.2.1 Information sur la sauvegarde

La documentation officielle de MongoDB parle déjà de différentes techniques de Sauvegarde :

- Sauvegarde en utilisant mongodump (Cf. la *section dédiée* (page 14)))
- Sauvegarde en utilisant Filesystem snapshots
- Sauvegarde par copie de disque.

Voir aussi :

Pour plus d'information, veuillez-vous référer à la documentation officielle : Monog dump [8] et Filesystem Snapshots [9].

La technique choisie par *VITAM* est celle de copie de disque pour plusieurs raisons :

- La taille de chaque shard est suffisamment grande. L'outil mongodump n'est pas conseillé dans ce cas de figure.
- La restauration prendra beaucoup de temps (restauration des données + création d'indexes)
- L'usage de mongodump impacte les performances

5.7.2.2 Procédure de sauvegarde

1. Repérer dans le replicaSet des instances mongo master (mongoc pour la conf ou mongod pour les shards)
2. Arrêter proprement mongo
3. Copie + zip du dossier db (avec le nommage qui permet d'identifier l'instance mongo) de chaque mongo master précédemment identifié
4. Stocker les zip dans un disque séparé et sécurisé

5.7.3 Restaurer MongoC et MongoD

Comme la sauvegarde, la documentation officielle parle aussi de la restauration d'un cluster shard mongo. Pour plus de détails, veuillez cliquer sur ce lien : Restore sharded cluster [10].

5.7.3.1 Procédure de restauration

1. Déployer *VITAM* qui va créer un cluster mongodb vide
2. Arrêter le cluster mongodb
3. Modifier la configuration de mongod et mongoc pour désactiver la replication et le sharding comme décrit dans la documentation officielle
4. Si vos instances mongo sont dans une zone réseau privée et sécurisée, vous pouvez aussi désactiver l'authentification en commentant la partie sécurité dans le fichier de conf
5. Copier et décompresser vers le dossier db de chaque instance mongo un fichier backup correspondant
   - Pour tous les mongoc, c'est le fichier zip backup mongoc qu'il faut mettre,
   - Pour chaque shard, il faut mettre le fichier zip du backup correspondant (pour chaque instance du replicaset du shard en question)
   - Le nom du zip doit mentionner l'instance mongo sur laquelle, on peut le restaurer
6. Démarrer toutes les instances mongo (en appliquant la modification des fichiers de configuration) une fois avoir copié et décompressé tous les backup sur toutes les instances
7. Dans chacune des instances
   (a) Si l'authentification est activée, il faut créer un `systemUser` (pré-requis : il faut un utilisateur ayant un role « root »)

https://docs.mongodb.com/manual/tutorial/backup-and-restore-tools/
https://docs.mongodb.com/manual/tutorial/backup-with-filesystem-snapshots/
https://docs.mongodb.com/manual/tutorial/restore-sharded-cluster/

```
use admin
// Authenticate as root user
db.auth("rootUser", "rootUserPassword")
// Create system user
db.createUser({user: "systmUser", pwd: "systemUserPassword", roles: [
↪"__system" ]})
// Authenticate as system user
db.auth("systmUser", "systemUserPassword")
```

(b) Supprimer la base de données `local`

```
// Drop local database
use local
db.dropDatabase()
```

(c) Pour les instances mongoc : Mettre à jour la collection `shards`

```
use config
// spécifier les shards pour chaque mongoc
// Example
db.shards.updateOne({ "_id" : "shard01"},  { $set : { "host" :
↪"shard01/shard01a:28018,shard01b:28018"}})
db.shards.updateOne({ "_id" : "shard02"},  { $set : { "host" :
↪"shard02/shard02a:28019,shard02b:28019"}})
db.shards.updateOne({ "_id" : "shard03"},  { $set : { "host" :
↪"shard03/shard03a:28020,shard03b:28020"}})
```

(d) Pour les instances mongod (les shards) : Mettre à jour la collection `system.version`

```
use admin
db.system.version.deleteOne( { "_id": "minOpTimeRecovery" } )
// spécifier les mongoc pour chaque shard
// Example
db.system.version.updateOne({ "_id" : "shardIdentity" },{ $set :{
↪"configsvrConnectionString" : "configserver/config01:28017,
↪config02:28017,config03:28017"}})
```

(e) Si vous avez crée un utilisateur ayant un role `__system` à l'étape (6.1), il faut donc le supprimer

```
// Remove system user
use admin
// Authenticate as root user
db.auth("rootUser", "rootUserPassword")
db.removeUser("systmeUser")
```

8. Arrêter mongodb et réactiver la replication et le sharding (et l'authentification si désactivée) dans la conf de chacune des instances
9. Démarrer Mongo
10. Activer les `replicaSet` pour chacun des mongoc et mongod (shards)

```
// Sur un des mongoc
> mongo --host {{ ip_service }} --port {{ mongodb.mongoc_port }} {{ vitam_
↪defaults.folder.root_path }}/app/mongoc/init-replica-config.js
// Pour chaque shards et sur un des shards d'un replicaset
> mongo --host {{ ip_service }} --port {{ mongodb.mongod_port }} {{ vitam_
↪defaults.folder.root_path }}/app/mongod/init-replica-config.js
```

11. Test de la restauration

- Un document accessible depuis un shards devrait être accessible depuis `mongos` (faire la requête de test sur chaque shard)
- Tester aussi les collections non shardées
- Il est conseillé de faire un `count` sur chacune des collections avant la sauvegarde pour vérifier lors de la restauration qu'on a bien les bons `count`.

Note : Si l'adresse ip et numéro de port de chacune des instances mongo du cluster recrée ne sont pas changés, alors les étapes 1, 2, 4, 8 et 10 sont suffisantes et le cluster mongodb devrait fonctionner sans problème.

Dans le cas ou la sécurité reste activée vous devez créer un utilisateur ayant un role « `__system` » et s'authentifier avec cet utilisateur pour pouvoir faire les modifications :

Prudence : Le pré-requis dans ce cas est d'avoir un utilisateur ayant un role « `root` » pour pouvoir créer un utilisateur ayant un rôle « `__system` »

L'ansiblerie *VITAM* déploie dans chacune des instances mongoc et mongod des scripts préparés restaure-mongoc.js et restaure-mongod.js respectivement

- {{ vitam_defaults.folder.root_path }}/app/mongoc/restaure-mongoc.js
- {{ vitam_defaults.folder.root_path }}/app/mongod/restaure-mongod.js

Toutes les informations sur les adresses ip et numéros de ports de toutes les instances du cluster mongodb sont automatiquement renseignés dans ces scripts

Pour exécuter ces deux scripts, il faut lancer la commande suivante que vous pouvez automatiser dans un playbook :

```
 // Sur mongoc
> mongo {{ ip_service }}:{{ mongodb.mongos_port }}/admin {{ mongo_credentials }} {{␣
↪vitam_defaults.folder.root_path }}/app/mongoc/restaure-mongoc.js
 // Sur mongod
> mongo {{ ip_service }}:{{ mongodb.mongos_port }}/admin {{ mongo_credentials }} {{␣
↪vitam_defaults.folder.root_path }}/app/mongod/restaure-mongod.js
```

5.7.4 Sauvegarde et restauration de l'offre froide

En plus de la procédure de backup et restauration décrite ci-dessus, pour *VITAM* ayant une offre de stockage froide, les fichiers backup zip sont stockés dans des bandes magnétiques.

5.7.4.1 Sauvegarde

La procédure de backup du mongo de l'offre froide est très importante, car, mongo joue le rôle d'un référentiel de tout ce qui est dans les bandes magnétiques.

Pour résumer, si on perd les données du cluster mongodb de l'offre froide, toutes les informations enregistrées sur les bandes magnétiques sont inutilisables.

C'est pour cette raison, que nous faisons, impérativement, au préalable :

- La sauvegarde du cluster mongodb de l'offre froide
- La sauvegarde est stockée sur bande magnétique.

5.7.4.1.1 Sauvegarde côté cluster mongodb de l'offre froide

Un playbook, ayant les tâches ci-dessous, a été mis en place pour faire un backup du mongodb de l'offre froide :

1. Détection des noeuds mongodb `master`
2. Arrêt de *VITAM*
3. Copie + ajout d'un fichier ayant des informations sur l'instance en cours + compression du dossier db de chaque instance `master`
4. Démarrer *VITAM*
5. Envoi des fichiers zip via CURL vers l'offre froide qui seront sauvegardés sur une bande magnétique

Pour exécuter le playbook :

Prudence : Le playbook ci-dessous est à exécuter uniquement sur un *VITAM* ayant une offre froide `**tapeLibrary**`

```
ansible-playbook -i environments/hosts.deployment ansible-vitam-exploitation/backup_
↪mongodb_tape_offer.yml --ask-vault-pass
```

5.7.4.1.2 Sauvegarde côté offre froide

Lors de l'envoi de fichier via une requête http vers l'offre froide, cette dernière va procéder comme suit :

- Réception du fichier zip dans une zone temporaire
- Copie du fichier dans une zone d'écriture sur bande magnétique
- Création d'un ordre spécifique pour écrire le fichier backup zip sur une bande magnétique ayant un tag `backup`
- Le worker qui va exécuter la tâche ayant l'ordre spécifique va écrire dans un fichier log `offer_tape_backup_DATE.log` les informations : `code de la bande magnétique`, `mongoc` ou `mongod (shard(i)`, `date`.

Note : Lors de la lecture depuis une bande magnétique, on aura un fichier zip mais sans connaitre son nom et son type. Si on perd le cluster mongodb, le fichier de log `offer_tape_backup_DATE.log` sera le seul moyen pour nous renseigner sur le nom du fichier sauvegardé et sur le code de la bande magnétique où il a été enregistré. Le nom `DATE-disk-mongod-shard01_.zip` qu'on récupère depuis le fichier log `offer_tape_backup_DATE.log` nous renseigne sur la date et le fait que ce soit un backup du `shard01`. Il ne peut donc être restauré que dans un `mongod` et non pas `mongoc`

5.7.4.2 Restauration

5.7.4.2.1 Restaurer côté offre froide

Sur l'offre froide, toutes les écritures de fichiers zip dans le cas de backup de mongodb de l'offre, sont tracées dans un fichier log `offer_tape_backup_DATE.log`

On peut facilement repérer des lignes de log ayant comme information :

- Le code de la bande magnétique sur laquelle est écrit le fichier
- Le nom du fichier de la forme `DATE-disk-mongod-shard01_.zip`

On saura donc dans quelle bande magnétique lire le fichier en question.

Avertissement : Il est fortement conseillé de copier ce fichier log `offer_tape_backup_DATE.log` dans un lieu sûr pour le besoin de restauration en cas de perte du site. Dans le cas contraire, on doit lire toutes les bandes magnétiques pour espérer retrouver les fichiers de backup.

Pour restaurer une date donnée

```
- On doit repérer dans le fichier log ``offer_tape_backup_DATE.log`` tous les␣
↪fichiers backup ``(mongoc et mongod)`` zip correspondant à cette date ainsi que les␣
↪bandes magnétiques sur lequelles on peut les lire
- Manuellement charger les bandes magnétiques sur une ``tape-library`` pour lire les␣
↪fichiers en question
- Renommer chacun des fichiers avec le nom adéquat (le nom se retrouve aussi à l
↪'intérieur du fichier zip dans un fichier descriptif)
- Copier et décompresser chacun de ces fichiers dans l'instance mongo correspondante␣
↪: Un fichier ayant un nom ``DATE-disk-mongod-shard01_.zip`` est à copier et à␣
↪décompresser dans toutes les instances mongo du shard ``shard01``
```

5.7.4.2.2 Restaurer côté cluster mongo de l'offre froide

Une fois tous les fichiers copiés et décompressés dans les instances mongo correspondantes, il faut suivre la procédure de restauration décrite ci-dessus paragraphe **Restaurer MongoC et MongoD**.

5.7.5 Cas de la base mongo certificates

Se référer à *Cas de la base mongo certificates* (page 16)

5.8 Gestion des profils de sécurité

La solution logicielle *VITAM* permet de gérer des profils de sécuité.

Le profil se base sur un contexte, lui-même basé sur une/des certificat(s).

Le processus d'installation met en place le profil de sécurité d'administration, qu'il est fortement recommandé de laisser « tel quel », car ce dernier est utilisé pour des actes d'exploitation.

Il n'existe actuellement pas de *playbook* permettant de rajouter des profils de sécurité.

5.8.1 Ajout de profils de sécurité

Avertissement : Cette version est encore en cours de mise en place et est susceptible d'évoluer.

5.8.1.1 Configuration

Un *playbook* d'exploitation permet de rajouter des profils de sécurité.

Sur la machine de déploiement, il est nécessaire de configurer le fichier `deployment/environments/group_vars/all/postinstall_param.yml`, dans la section `vitam_additional_securityprofiles`.

Exemple :

```
---

vitam_additional_securityprofiles:
  - name: applicative01
    identifier: spidentifier01 # manadatory, cannot be naither null nor blank
    hasFullAccess: false # true/false
    permissions: "null" # possible keypairs are stored in "modèle de données
↪"  ; for example : "contexts:read"
    contexts:
      - name: contextappli01
        identifier: myContextIdentifier # please leave blank if associated␣
↪tenants are "master" (if over release 9,related to existence of CONTEXT in␣
↪vitam_tenants_usage_external directive for tenant)
        status: ACTIVE # ACTIVE or INACTIVE
        enable_control: true # can be true or false , must be true if fine␣
↪permissions tuning (see above) ; if false, no permissions
        permissions: "[ { \"tenant\": 2, \"AccessContracts\": [], \
↪"IngestContracts\": [] }]" # you can specify different permissions for all␣
↪tenants that have to be associated to "[{perms1},{perms2}, ...]" where␣
↪permsX is  something like { \"tenant\": 2, \"AccessContracts\": [], \
↪"IngestContracts\": [] }
        certificates: ['cert1.crt'] # list of public certificates files␣
↪stored in {{inventory_dir}}/certs/client-external/clients/
```

Note : les certificats devraient être de type `external/${fichier crt}`.

5.8.1.2 Ajout des fichiers"crt"

Placer les certificats précédemment renseignés (fichiers crt) dans {{inventory_dir}}/certs/client-external/clients/external/.

5.8.1.3 Lancement du *playbook*

L'ajout des profils de sécurité se fait en lançant le *playbook* comme suit :

```
ansible-playbook -i environments/${inventaire} ansible-vitam-exploitation/
add_contexts.yml --ask-vault-pass
```

Prudence : Ce playbook ne sait gérer que le cas d'ajout de profils/contextes/... . Il convient de s'assurer au préalable que les champs *name* et *identifier* à ajouter n'existent pas déjà dans la solution logicielle *VITAM*.

5.8.1.4 Reconfiguration de VITAM

A l'issue de la bonne éxécution du *playbook*, il faut relancer un déploiement partiel de *VITAM* pour les groupes ansible `[hosts_ingest_external]` et `[hosts_access_external]`

5.8.1.4.1 Si utilisation de la PKI de tests

La procédure décrite ci-dessous est à appliquer dans le cas où la *PKI* de tests a été employée.

Avertissement : Si le certificat à ajouter a été généré avec une *CA* non-connue de VITAM, il faut ajouter au bon endroit la clé publique (se référer au *DIN* pour plus d'informations).

Prudence : Un fichier `crt` ne doit contenir qu'une clef publique

Ensuite, regénérer les *stores* Java avec les certificats supplémentaires (script `generate_stores.sh` ; se référer au *DIN* pour plus d'informations)

5.8.1.4.2 Cas d'une autre PKI

Mettre à jour les *stores* java avec les certificats supplémenataires à *truster*.

5.8.1.4.3 Application des *stores* mis à jour

Rejeu du déploiement en limitant aux groupes ansible `[hosts_ingest_external]` et `[hosts_access_external]` et avec le tag ansible `update_vitam_certificates`.

Exemples

```
ansible-playbook ansible-vitam/vitam.yml -i environments/<inventaire> -l hosts_ingest_
↪external,hosts_access_external -t update_vitam_certificates --vault-password-file␣
↪vault_pass.txt
ansible-playbook ansible-vitam/vitam.yml -i environments/<inventaire> -l hosts_ingest_
↪external,hosts_access_external -t update_vitam_certificates --ask-vault-pass
```

5.9 Batchs et traitements

5.9.1 Curator

Il existe des jobs Curator de :

- fermeture d'index
- suppression d'index fermés

Ces jobs sont lancés via `crontab` toutes les nuits.

5.9.2 *Timers* systemD

Tous les *timers* systemD décrits ci-dessous sont parématrables ; un comportement par défaut est appliqué. Se reporter à la procédure *Modifier la fréquence de lancement de certains timers systemD* (page 11) pour la bonne prise en compte du paramétrage attendu ou souhaité.

5.9.2.1 Sécurisation des journaux d'opérations

Un *timer* systemd a été mis au point pour réaliser ces actions :

- *vitam-traceability-operations* (page 60)

Ce *timer* est installé avec le composant `logbook`.

5.9.2.2 Sécurisation des cycles de vie

Des *timers* systemd ont été mis au point pour réaliser ces actions :

- *vitam-traceability-lfc-unit* (page 60)
- *vitam-traceability-lfc-objectgroup* (page 61)

Ces *timers* sont installés avec le composant `logbook`.

5.9.2.3 Sécurisation des offres de stockages

Des *timers* systemd ont été mis au point pour réaliser ces actions :

- *vitam-storage-log-backup* (page 59)
- *vitam-storage-log-traceability* (page 60)

Ces *timers* sont installés avec le composant `storage`.

5.9.2.4 Autres *timers*

Les *timers* suivants sont apportés par le composant `functional_administration`

```
vitam-create-accession-register-symbolic
vitam-functional-administration-accession-register-reconstruction
vitam-rule-management-audit
vitam-functional-administration-reconstruction
```

Les *timers* suivants sont apportés par le composant `metadata`

```
vitam-metadata-store-graph
vitam-metadata-reconstruction
vitam-metadata-computed-inherited-rules
vitam-metadata-purge-dip
vitam-metadata-purge-transfers-SIP
```

5.10 Sauvegarde des données graphe (Log shipping)

La sauvegarde des données graphe des métadonnées (UNIT/*GOT*) consiste à récupérer au fil de l'eau depuis la base de données (MongoDB) les données graphe par (UNIT/GOT) pour les stocker dans les offres de stockage.

Prudence : En cas de problème de sauvegarde des données *graphe*, on écrit dans le fichier log une `[Consistency Error]` qu'il conviendra de surveiller.

Avertissement : Si l'instance qui démarre le service de sauvegarde s'arrête, il faut lancer ce service de sauvegade dans une autre instance.

5.10.1 Déclenchement de la sauvegarde

La sauvegarde des données *graphe* est lancée via un *timer* systemd (*vitam-metadata-store-graph* (page 63)), qui démarre le service systemd associé.

- Le timer se lance chaque 30 minutes (par défaut, modifiable selon le besoin - se reporter à *Modifier la fréquence de lancement de certains timers systemD* (page 11) -)
- Le sauvegarde des données *graphe* se fait sur un intervalle de temps (depuis la dernière sauvegarde jusqu'au temps présent)
- Le fichier généré est un fichier au format `zip`, qui contient un ou plusieurs fichiers `JSON`. Ces fichiers `JSON` contiennent un tableau de données *graphe*.
- Le nom du fichier de la dernière sauvegarde contient les dates début et fin de sauvegarde. Ce nom est utilisé pour déterminer la dernière date de sauvegarde.
- La sauvegarde des UNIT est séparée de celle des *GOT* (Deux *containers* distincts dans chaque offre de stockage)

5.10.2 Reconstruction des données *graphe*

La reconstruction des données *graphe* se fait avec le même principe que la reconstruction des méta-données (UNIT/*GOT*) :

- Gérer l'offset de reconstruction des fichiers de sauvegarde des données graphe.
- Mettre à jour uniquement les données graphe. Si un document n'est pas trouvé, une création de ce document est faite et ne contiendra que les données du graphe.
- De même, la reconstruction des métadonnées ne modifie pas les données graphe potentiellement déjà existantes.
- Une purge est faite de tous les documents ayant uniquement les données graphe et qui sont vieux de (1 mois Configurable)
- Les documents ayant uniquement les données graphe ne sont pas indexés dans ElasticSearch.
- La reconstruction est séquentielle (D'abord les métadonnées UNIT/*GOT* ensuite leur graphe)

Voir aussi :

Se reporter à la procédure de *Reconstruction* (page 28) des métadonnées pour plus d'informations.

5.11 Recalcul des données graphe

Il est possible de recalculer les données du graphe en utilisant une requête *DSL*. En effet, dans le cadre de la procédure de *PRA*, il est nécessaire de pouvoir détecter les unités archivistiques ayant un graphe incohérent (construire le DSL requis) selon la procédure de déclenchement décrite ci-dessous.

Le recalcul de graphe permet de rétablir la cohérence des données *VITAM*.

Prudence : Cette procédure s'applique à partir de la version *VITAM* R8 (1.10.0).

Prudence : En cas de données de graphe incohérentes, le résultat des requêtes *DSL* sur les unités archivistiques pourra être incorrect et l'application des filtres de sécurité définis dans les contrats d'accès pourront ne pas être pris en compte.

5.11.1 Déclenchement

Le recalcul de *graphe* est déclenché par l'appel au point d'API porté par l'URL suivante sur le composant `metadata`

```
http://{{ ip_admin }}:{{ vitam.metadata.port_admin }}/metadata/v1/computegraph
```

Exemple d'appel à l'aide de curl :

```
curl -s -X POST -H "X-Tenant-Id: <tenant>" -H "Content-Type: application/json" --user
↪"${VITAM_ADMIN_AUTH}" --data @${CURRENT_DIR}/dslQuery.json ${URL}
```

Exemple de query DSL (dslQuery.json) :

```
{
  "$roots": [
    "aeaqaaaaaqhdytymabdeialenehzphiaaaeq",
    "aeaqaaaaaqhdytymabdeialenehzpbyaaajq"

... , "guid_n"
  ],
  "$query": [],
  "$projection": {}
}
```

La valeur utilisée pour la basic authentication sont issues du fichier vault.yaml et prennent la forme suivante

```
VITAM_ADMIN_AUTH={{ admin_basic_auth_user }}:{{ admin_basic_auth_password }}
```

- Le paramètre `adminUser` correspond à la valeur admin_basic_auth_user déclarée dans le fichier `vitam_security.yml`
- Le paramètre `adminPassword` correspond à la valeur admin_basic_auth_password déclarée dans le fichier `vault-vitam.yml`

Prudence : Si le *DSL* ne contient pas uniquement `$root`, alors la valeur du tenant positionnée dans le header `X-Tenant-Id` est essentielle car les index elasticsarch sont organisés par tenant.

5.12 Montée de version du fichier de signature de Siegfried

La solution logicielle *VITAM* utilise l'outil Siegfried pour la détection des formats des fichiers contenus dans les *SIP*. La nomenclature des formats est basée sur la norme « DROID » fournie par les archives nationales anglaises [11].

Ce fichier est régulièrement mis à jour et il est recommandé de procéder à sa mise en place dans *VITAM*.

Pour ce faire, il faut récupérer le fichier PRONOM [12] depuis les archives nationales anglaises et l'ajouter dans `deployment/environments`.

http://www.nationalarchives.gov.uk/
http://www.nationalarchives.gov.uk/information-management/manage-information/preserving-digital-records/droid/

Il faut ensuite modifier la valeur de la dierctive `droid_filename` dans le fichier `deployment/environments/group_vars/all/vitam_vars.yml` (avec le nom du fichier récupéré précédemment).

Enfin, il faut lancer le *playbook* ansible suivant depuis `deployment`

```
``ansible-playbook -i <fichier d'inventaire> ansible-vitam/roy_build_signature.yml --
↪ask-vault-pass``
```

ou bien, selon votre cas d'usage du *vault* ansible

```
``ansible-playbook -i <fichier d'inventaire> ansible-vitam/roy_build_signature.yml --
↪vault-password-file vault_pass.txt``
```

5.13 Griffins

Note : Nouveauté introduite en R9.

Afin de prendre en compte des considérations de réidentification et/ou préservation, la solution logicielle *VITAM* intègre désormais des *griffins* - greffons - pour réaliser les actions d'analyse, (ré)identification et préservation.

Comme décrit dans le *DIN*, le choix des *griffins* installés est défini dans le fichier `environments/group_vars/all/vitam-vars.yml` au niveau de la directive `vitam_griffins`.

Prudence : Cette version de la solution logicielle *VITAM* ne mettant pas encore en oeuvre de mesure d'isolation particulière des *griffins*, il est recommandé de veiller à ce que l'usage de chaque *griffin* soit en conformité avec la politique de sécurité de l'entité. Il est en particulier déconseillé d'utiliser un *griffin* qui utiliserait un outil externe qui n'est plus maintenu.

5.13.1 Ajout de nouveaux / mise à jour de *griffins*

Il est possible d'ajouter ou mettre à jour des *griffons* à une installation de la solution logicielle *VITAM*.

5.13.1.1 Ajout de *griffins*

Pour cela, il faut modifier le fichier `environments/group_vars/all/vitam-vars.yml` au niveau de la directive `vitam_griffins` par ajout du/des *griffon(s)* dans la liste.

Exemple d'ajout du greffon **vitam-un-nouveau-greffon-qui-est-necessaire**

```
vitam_griffins: ["vitam-imagemagick-griffin", "vitam-jhove-griffin","vitam-un-nouveau-
↪greffon-qui-est-necessaire"]
```

5.13.1.2 Mise à jour des *griffins*

Dans le cadre d'une montée de version des composants *griffins*, le *playbook* se charge de déployer les composants les plus à jour ; il n'est pas nécessaire de modifier la directive `vitam_griffins`.

Note : Ne pas oublier, sur les partitions associées, de mettre à jour, si nécessaire, l'adresse du dépôt de binaires.

5.13.1.3 Préparation du système

Il faut également prévoir de mettre à disposition sur le(s) dépôt(s) de binaires les packages d'installation correspondants. Le nom indiqué dans la liste doit correspondre au nom du package (format rpm ou deb selon la plateforme). Les packages d'installation associés doivent se situer dans un dépôt accessible et connu par la machine sur laquelle ils vont être installés.

5.13.1.4 Prise en compte technique par VITAM

Enfin, il suffit de relancer le *playbook* d'installation de *VITAM* avec, en fin de ligne, cette directive

```
--tags griffins
```

5.14 Reconstruction

La reconstruction consiste à recréer le contenu des bases de données (MongoDB-data, Elasticsearch-data) en cas de perte de l'une ou l'autre à partir des informations présentes dans les offres de stockage. Elle part du principe que le contenu des offres n'a pas été altéré.

Prudence : Dans cette version de la solution logicielle *VITAM*, la reconstruction nécessite de couper le service aux utilisateurs.

Prudence : Une reconstruction complète à partir des offres de stockage peut être extrêmement longue, et ne doit être envisagée qu'en dernier recours.

5.14.1 Procédure multi-sites

5.14.1.1 Cas du site primaire

La reconstruction se réalise de la manière suivante :

1. Arrêt de *VITAM* sur le site à reconstruire
   - Utiliser le playbook `ansible-vitam-exploitation/stop_vitam_timers.yml`
   - Utiliser le playbook `ansible-vitam-exploitation/stop_vitam.yml`

Il est indispensable de valider que tous les services *VITAM* (y compris les *timers* systemd) sont bien arrêtés

2. Purge des données (le cas échéant) stockées dans MongoDB-data, excepté les bases **identity**, **config** et **admin**.
3. Purge des données (le cas échéant) stockées dans Elasticsearch-data,
4. Reconfiguration et démarrage en tant que site secondaire :
   - Paramétrer la variable `primary_site` à `false` dans le fichier `environmenrs/group_vars/all/vitam_vars.yml` puis utiliser le playbook `ansible-vitam/vitam.yml`
5. Dès lors, l'accès utilisateur reste coupé, et l'intégralité des données est reconstruite progressivement
   - Le suivi de la reconstruction se fait en observant l'évolution de l'offset de reconstruction stocké dans MongoDB-data
   - Pour la release 8, la procédure est décrite dans la section « Recalcul des données graphe »

6. La collection `Offset` de la base de données `metadata` est créée et permet de suivre l'avancement de la reconstruction.
7. Une fois la reconstruction terminée (plus de modification dans la collection `Offset`), il convient de reconfigurer en tant que site primaire, puis redémarrer :
   - Paramétrer la directive `primary_site` à `true` puis utiliser le playbook `ansible-vitam/vitam.yml`

5.14.1.2 Cas du site secondaire

La reconstruction se réalise de la manière suivante :

1. Arrêt de *VITAM* sur le site à reconstruire
   - Utiliser le playbook `ansible-vitam-exploitation/stop_vitam_timers.yml`
   - Utiliser le playbook `ansible-vitam-exploitation/stop_vitam.yml`

Il est indispensable de valider que tous les services *VITAM* (y compris les *timers* systemd) sont bien arrêtés.

2. Purge des données (le cas échéant) stockées dans MongoDB-data, excepté les bases **identity**, **config** et **admin**.
3. Purge des données (le cas échéant) stockées dans Elasticsearch-data,
4. Redémarrage du site secondaire Vitam
   - Utiliser le playbook `ansible-vitam-exploitation/start_vitam.yml`
   - La prochaine itération de reconstruction au fil de l'eau redémarrera la reconstruction à partir du début
   - Attendre la fin de la reconstruction au fil de l'eau sur le site secondaire
     - Le suivi de la reconstruction se fait en observant l'évolution de l'offset de reconstruction stocké dans MongoDB-data
     - Pour la release 7 (version 1.4.x) il faut lancer le service dédié `vitam-metadata-graph-builder.service` sur le composant metadata pour recalculer le graphe des unités archivistiques et des groupes d'objets techniques n'ayant pas encore reconstruit leurs données graphe

5.14.2 Procédure mono-site

La procédure à appliquer est la même que la procédure du site primaire pour une installation multi-sites.

5.15 Plan de Reprise d'Activité (PRA)

Le *PRA* consiste à passer un site *VITAM* secondaire en site primaire après incident majeur survenu sur le site primaire (cas de l'indisponibilité complète du site primaire).

Note : Les actions en cours sur le site primaire sont perdues (versements non terminés, batchs etc.). L'incohérence des données sera traitée dans une version ultérieure de *VITAM*..

Cette section décrit des actions qui ne peuvent donc s'effectuer que si une installation multi-sites a été effectuée au préalable.

Cette section s'appuie sur les procédures décrites dans les chapitres suivants :

- Resynchronisation d'une offre à partir d'une autre offre (*Resynchronisation d'une offre* (page 32))
- Reconstruction des bases de données (MongoDB-data, Elasticsearch-data) en cas de perte de l'une ou l'autre, à partir des informations présentes dans les offres (*Reconstruction* (page 28))

5.15.1 Déclenchement

Avant le déclenchement de la procédure de *PRA*, le système fonctionne en mode multi-sites (primaire/secondaire). Le service est indisponible à la suite de la perte du site primaire.

Le déclenchement du *PRA* s'effecture selon la procédure suivante :

1. Vérifier que le site primaire est bien complètement arrêté
   - Il est indispensable de valider que tous les services VITAM (y compris les *timers* systemd) sont bien arrêtés
2. Attendre la fin de la reconstruction au fil de l'eau sur le site secondaire
   - Le suivi de la reconstruction se fait en observant l'évolution de l'offset de reconstruction stocké dans MongoDB-data.
   - Pour la release 7 (version 1.4.x) il faut lancer le service dédié `vitam-metadata-graph-builder.service` sur le composant metadata pour recalculer les données graphe des unités archivistiques et des groupes d'objets techniques n'ayant pas encore reconstruit leurs données graphe
3. Reconfigurer le site secondaire en site primaire :
   - Si le site secondaire était partiellement déployé, ne pas oublier de rajouter tous les composants requis pour un fonctionnement en site primaire
   - Attention à adapter la stratégie de stockage en fonction du mode d'utilisation choisi pour le site de secours :
     - Mode « lecture/écriture » : la stratégie de stockage doit être modifiée afin de limiter les écritures aux seules offres encore disponibles sur le site de secours (*Activation/désactivation d'une offre* (page 41))
     - Mode « lecture seule » (recherche et consultation avec profil de droits dédié) : la stratégie de stockage ne change pas. Seule une reconfiguration du site primaire initial en mode secondaire permettra le retour à un fonctionnement nominal (cf. ci-dessous)
   - Paramétrer la variable `primary_site` à `true` puis jouer le *playbook* `ansible-vitam/vitam.yml`
4. En lien avec le processus de reconstruction (cf. *Reconstruction* (page 28)), en cas de bascule sur le site secondaire, il sera préférable de purger les documents des Unit et ObjectGroup reconstruits mais ne contenant potentiellement que des données de graphe (cas de l'éliminitaion par exemple). Cette opération s'effectue avec la commande suivante :

```
curl -s -X DELETE -H "X-Tenant-Id: {{ vitam_tenant_admin }}" -H "Accept: application/
↪json" -H "Content-Type: application/json" --user "{{ admin_basic_auth_user }}:{{␣
↪admin_basic_auth_password }}" http://{{ ip_admin }}:{{ vitam.metadata.port_admin }}/
↪metadata/v1/purgeGraphOnlyDocuments/[UNIT | OBJECTGROUP | UNIT_OBJECTGROUP]
```

Après modification des accès pour les applications versantes (action infra. de type modification DNS, routage, conf etc.), le site secondaire peut alors être ouvert au service en tant que site primaire.

Le système fonctionne désormais en mode mono-site (primaire). Le service est de nouveau disponible sur le site de secours.

5.15.2 Retour en situation nominale

Le retour à la solution nominale s'effectue en deux étapes :

- Rétablissement du contenu du site primaire intial par reconfiguration temporaire en tant que site secondaire
- Retour à la configuration multi-sites initiale

Avertissement : Dans cette version, la resynchronisation partielle d'une offre de stockage n'étant pas supportée, le retour à la configuration multi-sites initiale nécessite de repartir d'offres vierges de toutes données sur le site à resynchroniser (on parle ici d'offre de remplacement)

5.15.2.1 Déclenchement

Avant déclenchement de la procédure de *PRA* inverse (retour en situation nominale), le système fonctionne en mode mono-site (primaire). Le service est disponible sur le site de secours.

Le déclenchement du *PRA* inverse s'effecture selon la procédure suivante :

- Vérifier que le site primaire initial est bien complètement arrêté
  - Il est indispensable de valider que tous les services VITAM (y compris les *timers* systemd) sont bien arrêtés
- Purger les données (le cas échéant) stockées dans MongoDB-data, excepté les bases **identity**, **config** et **admin**
- Purger les données (le cas échéant) stockées dans Elasticsearch-data
- Reconfigurer et démarrer le site primaire initial en tant que site secondaire :
  - Paramétrer la variable primary_site à false puis utiliser le playbook `ansible-vitam/vitam.yml`
  - Le mécanisme de reconstruction du contenu des bases de données (MongoDB-data, Elasticsearch-data) à partir des informations présentes dans les offres de stockage est actif (aucune donnée à resynchroniser à cette étape)
- Resynchroniser les offres de stockage à partir des offres du site de secours en se référant à la procédure suivante *Resynchronisation d'une offre* (page 32)
  - En fonction du mode d'utilisation choisi pour le site de secours :
    - Mode lecture/écriture : la stratégie de stockage du site de secours doit auparavant être modifiée afin de référencer de nouveau les offres du site primaire initial
    - Mode lecture seule : la stratégie de stockage ne change pas. Les offres du site primaire initial sont toujours connues du site de secours
- Le mécanisme de recontruction au fil de l'eau reconstruit progressivement le contenu des bases de données
  - Le suivi de la reconstruction se fait en observant l'évolution de l'offset de reconstruction stocké dans MongoDB data
  - Pour la release 7 (version 1.4.x) il faut lancer le service dédié `vitam-metadata-graph-builder.service` sur le composant metadata pour recalculer les données graphe des unités archivistiques et des groupes d'objets techniques n'ayant pas encore reconstruit leurs données graphe
- Une fois la reconstruction terminée, reconfiguration en tant que site primaire et démarrage :
  - Paramétrer la variable primary_site à true puis utiliser le playbook `ansible-vitam/vitam.yml`
- Reconfiguration et démarrage en tant que site secondaire du site de secours :

> **Avertissement :** Cette opération provoque une indisponibilité temporaire des principaux services *VITAM* (versement, gestion, recherche et consultation)

  - Paramétrer la variable primary_site à false puis utiliser le playbook `ansible-vitam/vitam.yml`

Après modification des accès pour les applications versantes (action infra. de type modification DNS, routage, conf etc.), le site primaire initial peut alors être de nouveau ouvert au service en tant que site primaire.

Le système fonctionne désormais de nouveau en mode multi-sites (primaire/secondaire). Le service est de nouveau disponible sur le site primaire initial.

5.16 Resynchronisation d'une offre

Une offre de stockage peut être désynchronisée par rapport à une autre à la suite d'une indisponibilité plus ou moins longue voire totale de l'offre (*crash* majeur du système, panne matérielle etc.) ou bien encore à la suite d'une mise en maintenance programmée.

Le mécanisme de resynchronisation d'une offre par rapport à une autre nécessite une intervention d'exploitation manuelle permettant de remédier à la perte de données dans le système.

Note : En cas d'indisponibilité d'une offre, le processus d'entrée d'un *SIP* n'étant réussi que si et seulement si toutes les offres de stockage definies dans la strategie sont accessibles, et que tous les fichiers sont bien copiés sur la totalité de ces offres, il sera nécessaire de désactiver l'offre (cf. chapitre *Activation/désactivation d'une offre* (page 41)) afin de permettre à nouveau les entrées de *SIP* (ingest/versement).

Prudence : Il est recommandé de procéder à un audit d'intégrité dans le cadre d'opérations techniques ciblées, telles que l'évolution de la stratégie de stockage, un changement de stockage.

5.16.1 Cas de l'ajout d'une nouvelle offre

Avertissement : Lors de l'ajout d'une nouvelle offre (portant un nouvel identifiant d'offre), les métadonnées des AU / GOT existants ne seront pas mis à jour avec l'information sur la nouvelle stratégie de stockage utilisée. L'ajout d'un mécanisme de mise à jour / propagation des métadonnées est prévu dans une version ultérieure. Lors de l'ajout d'une offre en remplacement d'une précédente offre, l'intégrité des métadonnées sera garantie en conservant le même identifiant d'offre.

L'ajout d'une nouvelle offre de stockage requiert le déploiement des applicatifs *VITAM* associés selon la procédure suivante :

- Editer le fichier de configuration de l'inventaire de déploiement (généralement, fichier `hosts`) afin d'ajouter les nouveaux serveurs portant les composants à déployer (fonction de la topologie de déploiement retenue) :

```
[hosts_storage_offer_default]
hostname-new-storage-offer      offer_conf=<offer-z>

[hosts_mongos_offer]
hostname-new-mongos-offer       offer_conf=<offer-z>

[hosts_mongoc_offer]
hostname-new-mongoc-offer       offer_conf=<offer-z>

[hosts_mongod_offer]
hostname-new-mongod-offer       offer_conf=<offer-z>
```

- Editer le fichier de configuration de la strategie de stockage `offer_opts.yml` afin d'ajouter la nouvelle offre :

```
vitam_strategy:
- name: <offer-x>
    referent: true
# <offer-z> is the new offer
```

(suite sur la page suivante)

(suite de la page précédente)

```
- name: <offer-z>
    referent: false

vitam_offers:
    <offer-x>:
        provider: filesystem
    # <offer-z> is the new offer
    <offer-z>:
        provider: filesystem
```

Si la nouvelle offre est utilisée dans une stratégie additionnelle (`other_strategies`), la modification sera la suivante :

```
other_strategies:
    metadata:
    - name: <offer-x>
        referent: false
    # <offer-z> is the new offer
    - name: <offer-z>
        referent: false

vitam_offers:
    <offer-x>:
        provider: filesystem
    # <offer-z> is the new offer
    <offer-z>:
        provider: filesystem
```

- Editer le fichier de déclaration des secrets généraux `vault-vitam.yml` afin d'y ajouter les secrets associés à la nouvelle offre :

```
mongodb:
    <offer-z>:
        passphrase: <passphrase>
        admin:
          user: <admin-user>
          password: <admin-password>
        localadmin:
          user: <localadmin-user>
          password: <localadmin-password>
        offer:
          user: <offer-user>
          password: <offer-password>
```

- Exécuter la commande suivante afin de déployer les nouveaux composants storage-offer, mongos-offer, mongoc-offer, mongod-offer :

Note : On considère que les étapes de génération des *hostvars*, de génération des magasins de certificats et de mise en place des repositories *VITAM* ont été réalisées au préalable pour les serveurs concernées (se référer aux sections du *DIN* correspondantes).

```
ansible-playbook -i environments/<hosts> -l "hostname-new-storage-offer,hostname-new-
↪mongos-offer,hostname-new-mongoc-offer,hostname-new-mongod-offer" ansible-vitam/
↪vitam.yml --ask-vault-pass
```

La nouvelle offre doit ensuite être déclarée dans la stratégie de stokage par reconfiguration du moteur de stockage selon la procédure suivante :

Avertissement : Cette opération provoque une indisponibilité temporaire des principaux services *VITAM* (versement, gestion, recherche et consultation)

- Exécuter la commande suivante afin de reconfigurer le composant storage-engine :

```
ansible-playbook -i environments/<hosts> -l hosts_storage_engine ansible-
↪vitam/vitam.yml --ask-vault-pass --tags update_vitam_configuration
```

5.16.2 Cas de la resynchronisation d'une offre temporairement indisponible

La resynchronisation d'une offre à partir du contenu d'une autre offre s'effectue en suivant la procédure suivante :

Note : Cette procédure n'impacte pas les services *VITAM*. Le mécanisme de reconstruction du contenu des bases de données (MongoDB-data, Elasticsearch-data) à partir des informations présentes dans les offres de stockage fonctionne de manière concurrente au mécanisme de resynchronisation.

- Exécuter la commande suivante afin de resynchroniser la nouvelle offre vis-à-vis de l'offre (des offres) source(s) :

```
curl -v -X POST -u adminUser:adminPassword --header 'content-type:␣
↪application/json' --header 'accept: application/json' http://
↪<storageengine>:29102/storage/v1/offerSync --data '
{
    "sourceOffer": "<offer-x>.service.consul",
    "targetOffer": "<offer-z>.service.consul",
    "strategyId": <strategyId>,
    "container": <datatype>,
    "tenantId": <tenantId>
}'
```

  - Le paramètre `adminUser` correspond à la valeur `admin_basic_auth_user` déclarée dans le fichier `vitam_security.yml`
  - Le paramètre `adminPassword` correspond à la valeur `admin_basic_auth_password` déclarée dans le fichier `vault-vitam.yml`
  - Le paramètre `sourceOffer` correspond à l'**id** de l'offre source utilisée pour la resynchronisation de la nouvelle offre
  - Le paramètre `targetOffer` correspond à l'**id** de l'offre à resynchroniser
  - Le paramètre `strategyId` correspond à la stratégie des offres source et cible
  - le paramètre `tenantId` correspond au tenant sur lequel appliquer la synchronisation
  - Le paramètre `container` correspond à un élément datatype de la liste suivante :

```
"units"
"objects"
"objectgroups"
"logbooks"
"reports"
"manifests"
"profiles"
"storagelog"
```

(suite sur la page suivante)

(suite de la page précédente)

```
"storageaccesslog"
"storagetraceability"
"rules"
"dip"
"agencies"
"backup"
"backupoperations"
"unitgraph"
"objectgroupgraph"
"distributionreports"
"accessionregistersdetail"
"accessionregisterssymbolic"
```

- Suivre les journaux de la resynchronisation dans les logs du composant storage offer avec la commande suivante :

```
tail -F /vitam/log/storage/storage_offer_sync.\*.log
```

- Vérifier l'état d'exécution de la synchronisation via la commande (peut être scriptée) :

```
curl -v -X HEAD -i -u adminUser:adminPassword http://<storageengine>
↪:29102/storage/v1/offerSync
```

  L'entête `Running` indique l'état d'exécution de processus de synchronisation.

- Vérifier le détail d'exécution de la synchronisation via la commande :

```
curl -v -X GET -u adminUser:adminPassword http://<storageengine>:29102/
↪storage/v1/offerSync
```

- En cas de resynchronisation partielle d'une offre, il est possible d'exécuter le processus de resynchronisation à partir d'un `offset` :

```
curl -v -X POST -u adminUser:adminPassword --header 'content-type:␣
↪application/json' --header 'accept: application/json' http://
↪<storageengine>:29102/storage/v1/offerSync --data '
{
    "sourceOffer": "<offer-x>.<consul_domain>",
    "targetOffer": "<offer-z>.<consul_domain>",
    "strategyId": <strategyId>,
    "offset": <offset>,
    "container": <datatype>,
    "tenantId": <tenantId>
}'
```

  Où <datatype> doit prendre les valeurs suivantes :

```
"units"
"objects"
"objectgroups"
"logbooks"
"reports"
"manifests"
"profiles"
"storagelog"
"storageaccesslog"
"storagetraceability"
```

(suite sur la page suivante)

(suite de la page précédente)

```
"rules"
"dip"
"agencies"
"backup"
"backupoperations"
"unitgraph"
"objectgroupgraph"
"distributionreports"
"accessionregistersdetail"
"accessionregisterssymbolic"
```

- Le paramètre `offset` correspond à la valeur du dernier `offset` observé dans les logs du composant storage offer (cas d'une reprise suite à interruption ou échec de la procédure de resynchronisation). Le paramètre `offset` peut égaemment être déterminé via les enregistrements de la collection `OfferLog` (database `offer`) depuis la base MongoDB associée à l'offre à resynchroniser (cas d'une panne ou d'une mise en maintenance programmée à une date précise).

5.17 Procédure d'exploitation suite à la création ou la modification d'une ontologie

Au préalable à la création ou à la modification d'une ontologie, les index Elasticsearch correspondant aux ontologies doivent être créés ou mis à jour.

5.17.1 Création d'une ontologie

Suite à la création d'une nouvelle ontologie, les index Elasticsearch doivent être mis à jour selon la procédure suivante :

- Dans le cas d'une création, il suffit de créer un nouveau mapping dans les index concernés.

Ex : Ajout d'une propriété Licence dans tous les index unit (unit* signifiant tous les index unit unit_0, unit_1 etc . . .)

Commandes à lancer sur une des partitions hébergeant le cluster elasticsearch « data » :

```
curl -XPUT "http://localhost:9200/unit*/_mapping/typeunique?update_all_types" -d'
{
      "properties": {
        "Licence": {
          "type": "text"
        }
      }
}'
```

Pour verifier sur un ou tous les index `unit` :

```
curl -XGET "http://localhost:9200/unit_0/_mapping/?pretty=true"
```

```
curl -XGET "http://localhost:9200/unit*/_mapping/?pretty=true"
```

5.17.2 Changement de type d'une ontologie existante

Dans ce cas, le changement de type dans elasticsearch n'est pas possible. Il faut donc créer un nouvel index Elastic-Search avec un nouveau mapping, puis reindexer l'ancien index dans ce dernier.

On récupère d'abord l'ancien index

```
curl -XGET 'localhost:9200/unit_1/_mapping?pretty=true'
```

On créé un fichier json et on y copie les données obtenues (ne conserver que la balise « mappings » : { ... } et son contenu). On modifie le mapping en changeant le type des propriétés choisies. On créé un nouvel index on lui passant en paramètre le fichier du nouveau mapping .

```
curl -XPUT "http://localhost:9200/new_unit_1" -H 'Content-Type: application/json' -d␣
↪@newmapping.json
```

Verifier l'index :

```
curl -XGET 'localhost:9200/new_unit_1/_mapping/'
```

On reindexe unit_1 vers le nouvel index new_unit_1

```
curl -XPOST 'localhost:9200/_reindex' -H 'Content-Type:application/json' -d '{
        "source" : {
                "index" : "unit_1"
        },
        "dest" : {
                "index" : "new_unit_1",
                "version_type": "external"
        }
}'
```

On efface l'alias de l'ancien index unit_1

```
curl -XDELETE 'localhost:9200/unit_1/_alias/unit_1'
```

et on l'affecte au nouvel index new_unit_1

```
curl -XPUT 'localhost:9200/new_unit_1/_alias/unit_1'
```

Avertissement : les index elasticsearch de *VITAM* sont créés par tenant. Il faudra refaire l'opération ci-dessus pour chaque tenant.

Avertissement : en cas de reindexation des index elasticsearch par le service *REST* de *VITAM*, les données sont réindexées suivant le mapping initial. Les nouveaux mappings ne seront donc pas pris en compte. Ce comportement sera modifié dans le futur.

5.18 Procédure d'exploitation pour la mise en pause forcée d'une opération

Pour permettre le traitement non-concurrent de certaines opérations (ingest et reclassement en particulier), il est possible de pouvoir forcer la mise en pause à la réception d'opérations (toutes ou seulement d'un type donné, sur tous les tenants ou un tenant donné en particulier).

Concrètement, elle permet de forcer le mode « pas à pas » pour toutes ou un type donné seulement d'opérations, sur l'ensemble des tenants ou sur un tenant donné seulement.

5.18.1 Mise en pause forcée

La mise en pause forcée est déclenchée par l'appel au point d'API porté par le composant `access-external` à l'URL suivante : `http://{{ ip_service }}:{{ vitam.accessexternal.port_service }}/admin-external/v1/forcepause`

Exemple d'appel à l'aide de curl :

```
curl -X POST -k
--key vitam-vitam_1.key
--cert vitam-vitam_1.pem
``https://{{ ip_service }}:{{ vitam.accessexternal.port_service }}/admin-external/v1/
↪forcepause``
-H 'X-Tenant-Id: 0'
-H 'X-Access-Contract-Id: ContratTNR'
-H 'Content-Type: application/json;charset=UTF-8'
-H 'Accept: application/json'
--data-binary '{"type" : "INGEST", "tenant" : "0"}'
--compressed
```

Exemple de Json pour mettre une pause sur le processus d'ingest pour le tenant 0 :

```
'{"type" : "INGEST", "tenant" : "0"}'
```

Exemple de Json pour mettre une pause sur tous les processus pour le tenant 0 :

```
'{"tenant" : "0"}'
```

Exemple de Json pour mettre une pause sur tous les processus pour tous les tenants :

```
'{"pauseAll":true}'
```

5.18.2 Sortie de la mise en pause forcée

La sortie de mise en pause forcée est déclenchée par l'appel au point d'API porté par le composant `access-external` à l'URL suivante : `http://{{ ip_service }}:{{ vitam.accessexternal.port_service }}/admin-external/v1/removeforcepause`

Exemple d'appel à l'aide de curl :

```
curl -X POST -k
--key vitam-vitam_1.key
--cert vitam-vitam_1.pem
``https://{{ ip_service }}:{{ vitam.accessexternal.port_service }}/admin-external/v1/
↪removeforcepause``
-H 'X-Tenant-Id: 0'
-H 'X-Access-Contract-Id: ContratTNR'
-H 'Content-Type: application/json;charset=UTF-8'
-H 'Accept: application/json'
--data-binary '{"type" : "INGEST", "tenant" : "0"}'
--compressed
```

Exemple de Json pour sortir de la mise en pause sur le processus d'ingest pour le tenant 0 :

```
'{"type" : "INGEST", "tenant" : "0"}'
```

Exemple de Json pour sortir de la mise en pause sur tous les processus pour le tenant 0 :

```
'{"tenant" : "0"}'
```

Exemple de Json pour sortir de la mise en pause sur tous les processus pour tous les tenants :

```
'{"pauseAll":false}'
```

Avertissement : Les états de mise en pause ne sont pas sauvegardés. En cas de redémarrage des applications (en particulier le composant access-external), ces états sont perdus.

5.19 Réindexation

Cette procédure consiste à réindexer le contenu des bases de données Elasticsearch-data (cluster d'indexation dédié aux données métier) en cas de perte ou d'inconsistence de données, à partir des informations présentes dans les bases de données MongoDB-data (replicaset MongoDB stockant les données métier de Vitam). Elle part du principe que le contenu des collections MongoDB-data n'a pas été altéré et que les différents index Elasticsearch-data sont toujours existants.

5.19.1 Déclenchement

La réindexation se déclenche de la manière suivante :

```
ansible-playbook ansible-vitam-exploitation/reindex_es_data.yml -i environments/$
↪{fichier_d_inventaire} --ask-vault-pass
```

Ce *playbook* s'assure que le composant `vitam-functional-administration` est démarré, puis procède à la réindexation et au *re-aliasing* (bascule sur le nouvel index) des collections suivantes :

- unit
- logbookoperation
- objectgroup
- securityprofile
- context
- ontology
- ingestcontract
- agencies
- accessionregisterdetail
- archiveunitprofile
- accessionregistersummary
- accesscontract
- fileformat
- filerules
- profile
- griffin
- preservationscenario
- managementcontract

Note : La réindexation peut s'opérer au besoin sur uniquement l'une des collections ci-dessus en spécifiant l'option –tags <collection> à l'exécution de la commande ansible.

Prudence : La réindexation de la collection griffin n'est pas utilisable dans cette version (bug 5762).

Prudence : La purge des anciens index n'est pas réalisée par cette procédure scriptée et est laissée à la charge de l'exploitant.

5.20 Procédure d'exploitation pour la révocation des certificats SIA et Personae

Cette section fait référence au chapitre *Intégration d'une application externe dans Vitam* (page 185).

La version 1.10.0 (« R8 ») introduit une nouvelle fonctionnalité permettant la révocation des certificats *SIA* et *Personae* afin d'empecher des accès non autorisés aux *API* de la solution logicielle *VITAM* (vérification dans la couche https des *CRL*).

Le fonctionnement de la validation des certifcats de la solution logicielle *VITAM* *SIA* et *Personae* par *CRL* est le suivant :

- L'administrateur transmet à la solution logicielle *VITAM* le *CRL* d'un *CA* qui a émis le certificat présent dans la solution logicielle *VITAM*, via le point d'API suivant

```
http://{{ hosts_security_internal }}:{{ vitam.security_internal.port_admin }}/v1/
↪api/crl
```

Prudence : La CRL fournie doit être obligatoirement au format DER (cf. http://www.ietf.org/rfc/rfc3280.txt »>RFC 3280 : *Internet X.509 Public Key Infrastructure Certificate and CRL Profile*)

Exemple :

```
curl -v -X POST -u {{ admin_basic_auth_user }}:{{ admin_basic_auth_password }}  http:/
↪/{{ hosts_security_internal }}:{{vitam.security_internal.port_admin}}/v1/api/crl -H
↪'Content-Type: application/octet-stream' --data-binary @/path/to/crl/my.crl
```

Le paramètre `adminUser` correspond à la valeur `admin_basic_auth_user` déclarée dans le fichier `vitam_security.yml`

Le paramètre `adminPassword` correspond à la valeur `admin_basic_auth_password` déclarée dans le fichier `vault-vitam.yml`

- Le système va contrôler tous les certificats (collections `identity.Certificate` et `identity.PersonalCertificate`) émis par le *IssuerDN* correspondant à la *CRL*, en vérifiant si ces derniers sont révoqués ou non. Si c'est le cas, alors la solution logicielle *VITAM* positionne le statut du certificat révoqué à **REVOKED**. Cela a pour conséquence le rejet de tout accès aux *API* *VITAM* avec utilisation du certificat révoqué (les filtres de sécurité émettront des exceptions dans les journaux de *log*).
- Une alerte de sécurité est émise dans les journaux en cas de révocation.

5.21 Activation/désactivation d'une offre

Prudence : Ne pas oublier, en cas de désactivation d'une offre considérée référente par *VITAM*, de déclarer une autre offre (contenant les données) comme nouvelle référente (modifications à apporter dans `deployment/environments/group_vars/all/offers_opts.yml` par la directive `referent: true`).

Lors de la détection de la perte d'une offre de stockage ou lors d'une maintenance programmée sur celle-ci, afin de permettre à nouveau les versements, l'offre pourra être désactivée.

La désactivation d'une offre s'effectue selon la procédure suivante :

- Exécuter la commande suivante afin de déscativer l'offre :

```
ansible-playbook -i environments/<hosts> ansible-vitam-exploitation/
↪activate_vitam_offer.yml --ask-vault-pass
```

Rentrer (entre quotes) le nom de l'offre (parmi les offres listées par la question) à activer/désactiver en réponse à la question suivante :

```
Which offer do you want to disable ?
   nom-offre-1
   nom-offre-2
   ...
   nom-offre-n
  : "nom-offre-x"
```

Choisir l'action à réaliser en répondant à la question suivante :

```
Do you want to enable (yes) or to disable (no) the chosen offer ?
Answer with ('yes'|'no') : no
```

Signification de la réponse :

- **yes** : active l'offre choisie
- **no** : désactive l'offre choisie

Avertissement : Cette opération provoque une indisponibilité complète de *VITAM*

- Exécuter la commande suivante afin de reconfigurer le composant storage-engine :

```
ansible-playbook -i environments/<hosts> -l hosts_storage_engine ansible-
↪vitam/vitam.yml --ask-vault-pass --tags update_vitam_configuration
```

Avertissement : Cette opération provoque une indisponibilité temporaire des principaux services *VITAM* (versement, gestion, recherche et consultation)

5.22 Nettoyage d'un environnement

Avertissement : La procédure suivante ne peut être appliquée QUE sur un environnement de recette et NE DOIT PAS être appliquée sur un environneent de production.

Un *playbook* ansible de nettoyage d'un environnement est fourni et permet de purger un environnement afin de le réinitialiser à son état presque initial.

Le nettoyage d'un environnement s'effectue selon la procédure suivante :

- Exécuter la commande suivante afin de nettoyer l'environnement (offres de stockage, workspace et bases de données) :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
↪cleaning.yml --ask-vault-pass
```

  Ou, si un vault-pass-word-file est utilisé

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
↪cleaning.yml --vault-password-file vault_pass.txt
```

En détails, le nettoyage d'un environnement va exécuter la liste des actions suivantes :

1. Arrêt des modules externes, afin que Vitam ne puisse plus accepter de demandes provenant de l'extérieur.
2. Purge des collections MongoDB :

- `LogbookLifeCycleObjectGroup`
- `LogbookLifeCycleObjectGroupInProcess`
- `LogbookLifeCycleUnit`
- `LogbookLifeCycleUnitInProcess`
- `LogbookOperation`
- `AccessionRegisterDetail`
- `AccessionRegisterSummary`
- `AccessionRegisterSymbolic`
- `ObjectGroup`
- `Unit`
- `EliminationActionObjectGroup`
- `EliminationActionUnit`
- `PreservationReport`

3. Réindexation des collections à indexer et purgées en phase 2 : le but étant d'obtenir en fin de traitement des indexes vides (ne contenant aucun document).

- `LogbookOperation`, `ObjectGroup` & `Unit` : une réindexation sera effectuée pour chaque tenant configuré (ex : unit_0_20190130_104530, unit_1_20190130_104540, etc...).
- `AccessionRegisterDetail`, `AccessionRegisterSummary`, `AccessionRegisterSymbolic` : 3 nouveaux indexes seront créés au total pour les 3 collections (ex : `accessionregistersymbolic_20190121_133507`, `accessionregistersummary_20190121_133503` & `accessionregisterdetail_20190121_133505`)

4. Nettoyage des offres de stockage. Selon la configuration de l'environnement, le script est en charge de nettoyer chaque offre de stockage configurée :

- Pour chaque tenant configuré dans Vitam, le script va supprimer tous les sous-containers ainsi que leurs contenus exceptés : « backup » (contenant les référentiels intégrés dans Vitam) et « rules ».
- Pour l'offre File-System, chaque sous-containers supprimé et vidé de son contenu, sera recréé à vide (ex : `int_0_accessionregisterdetail`)

5. Nettoyage du *workspace* : le contenu des objets contenus dans le workspace sera purgé également.

Avertissement : Pour le moment, seules les offres de stockage S3 et File-System sont prises en compte dans la purge des offres de stockage.

Prudence : Les nettoyage des offres Swift ne fonctionne que dans le cas d'une installation de la solution logicielle *VITAM* en environnement CentOS.

Avertissement : Cette opération provoque une indisponibilité complète de *VITAM*

5.22.1 Etat des lieux après purge

Après le passage du script, l'environnement est purgé :

- Les référentiels sont toujours présents (Contrats d'accès, contrats d'entrée, rêgles de gestion, etc...).
- L'environnement est disponible et utilisable : les modules externes sont accessibles (IngestExternal et AccessExternal).
- Les offres de stockage sont vidées, à l'exception des backups des référentiels.
- La cohérence entre MongoDB et ElasticSearch est assurée. La plupart des collections sont vidées, et les indexes E/S associés ne contiennent aucun document.
- Le workspace est purgé, aucune opération n'est en cours et ne peut être relancée.

5.22.2 Limitations

Le fait de purger les journaux et non les référentiels provoquera une incohérence de la plate-forme vis-à-vis de la norme **NFZ-42020** (suppression des logs d'imports de référentiels, mais présence de ceux-ci).

CHAPITRE 6

Suivi de l'état du système

6.1 Veille et patchs sécurité

Les éléments d'infrastructure suivants sont particulièrement sensibles pour la sécurité de la solution logicielle *VITAM* et nécessitent d'être intégrés à la veille sécurité du système :

- Runtime Java (OpenJDK 8)

6.2 Métriques

La solution logicielle *VITAM* intègre une solution de *monitoring* des applications à l'aide de métriques. L'exploitant peut, s'il le souhaite, changer la configuration des remontées de métriques, ou bien utiliser celle par défaut proposée dans *VITAM*.

6.2.1 Configuration

6.2.1.1 Activation/désactivation

Il est possible de définir les composants sur lesquels activer/désactiver ces métriques. Pour cela, il faut éditer le fichier `deployment/environments/group_vars/all/vitam_vars.yml` et définir, pour chaque composant,la valeur de la directive `metrics_enabled` (true/false).

A l'issue, lancer la commande de dépploiement (se reporter au *DIN*).

6.2.1.2 Registres

Par défaut, 3 registres de métriques sont créés pour toutes les applications VITAM :

- les métriques de Jersey
- les métriques de la JVM (Java Virtual Machine)

- les métriques « métier »

JERSEY : Les métriques Jersey correspondent à 3 métriques, des *Timers*, des *Meters*, et des *ExceptionMeters* qui vont être enregistrées pour chaque URI des API Rest de VITAM.

- Les *Meters* font office de compteurs. Ils sont incrémentés de 1 chaque fois qu'une URI est requêtée.
- Les *Timers* font office de chronomètres. Ils chronomètrent le temps de réponse d'une URI chaque fois que celle-ci est requêtée.
- Les *ExceptionMeters* font office de compteurs. Ils sont incrémentés de 1 chaque fois qu'une URI soulève une Exception dans le code.

JVM : Les métriques JVM correspondent à des *Gauges* qui enregistrent des valeurs de ressources système utilisées par la Java Virtual Machine pour chaque application VITAM.

BUSINESS : Les métriques métiers correspondent à des métriques de n'importe quel type qui peuvent remonter toute donnée considérée utile dans une application VITAM.

6.2.1.3 Reporters

Par défaut, 2 reporters de métriques sont disponibles pour les applications VITAM. Les reporters de métriques sont en charge de collecter les valeurs des métriques à des intervalles réguliers.

LogBack : le reporter LogBack affiche les valeurs des métriques dans LogBack.

ELASTICSEARCH : le reporter ElasticSearch sauvegarde les valeurs des métriques dans une base de données ElasticSearch qui peut être configurée dans le fichier de configuration.

6.2.1.4 Fichier de configuration

Le fichier de configuration des métriques est situé dans `/vitam/conf/<service_id>/vitam.metrics.conf`. Ce fichier contient la documentation nécessaire pour configurer correctement les métriques. Une description des clés YAML y est disponible.

6.2.2 Métier

Aucun métrique n'a encore été défini à ce stade du projet.

6.2.3 Métriques techniques

6.2.3.1 Métriques système critiques

Aucun métrique n'a encore défini à ce stade du projet.

6.2.3.2 Indicateurs de SLA

Aucun indicateur n'a encore défini à ce stade du projet.

6.2.3.3 Indicateurs de performance

Aucun indicateur n'a encore défini à ce stade du projet.

6.2.4 Visualisation

Si un reporter de type **ElasticSearch** est configuré, alors les métriques peuvent être visualisées via l'application web Kibana [13].

L'application Kibana comporte 4 sections qui seront développées :

- **Discover**
- **Visualize**
- **Dashboards**
- **Settings**

Néanmoins si vous souhaitez travailler avec Kibana, il est judicieux de consulter la documentation officielle. Celle-ci n'ayant pour but qu'une présentation sommaire de l'outil.

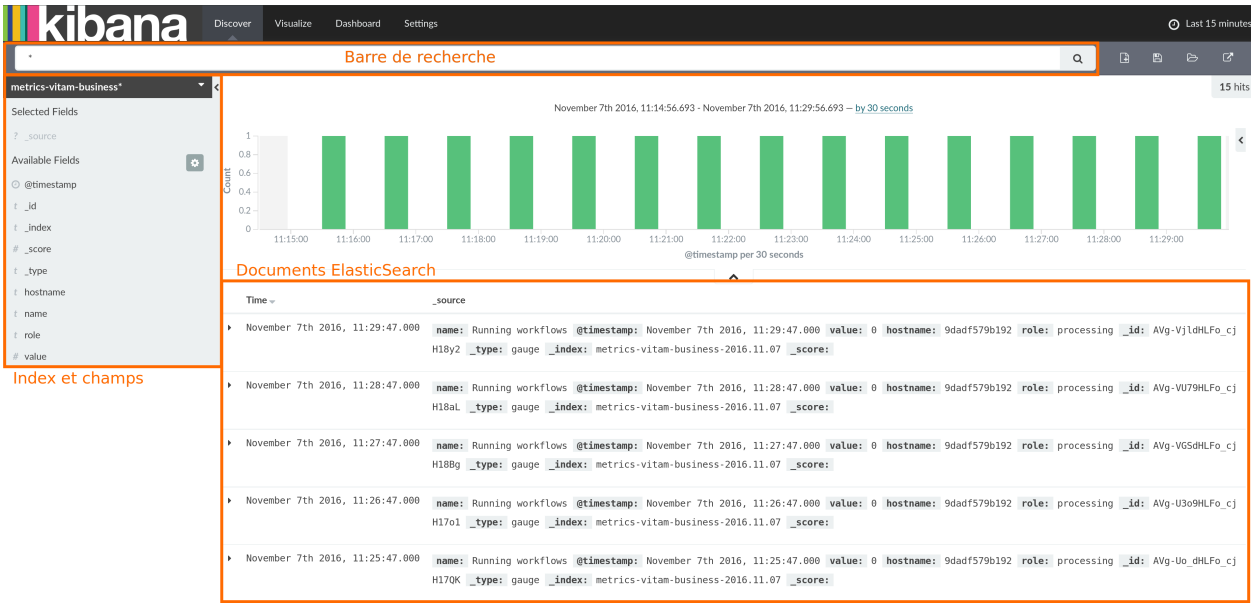
Voir aussi :

Documentation officielle de Kibana [14]

6.2.4.1 Discover

La section **Discover** permet de consulter rapidement les données présentes dans un index d'ElasticSearch. Pour cela, il suffit de séléctionner un index dans la barre latérale gauche, de choisir les champs que l'on souhaite consulter (optionnel) et les données apparraissent triées par ordre chronologique décroissant.

Il est possible d'effectuer des recherches poussées sur les documents, comme des expressions régulières, grâce à la barre de recherche en haut de la page. Une fois la recherche executée, il peut être utile de la sauvegarder afin de la réutiliser pour des visualisations.



6.2.4.2 Visualize

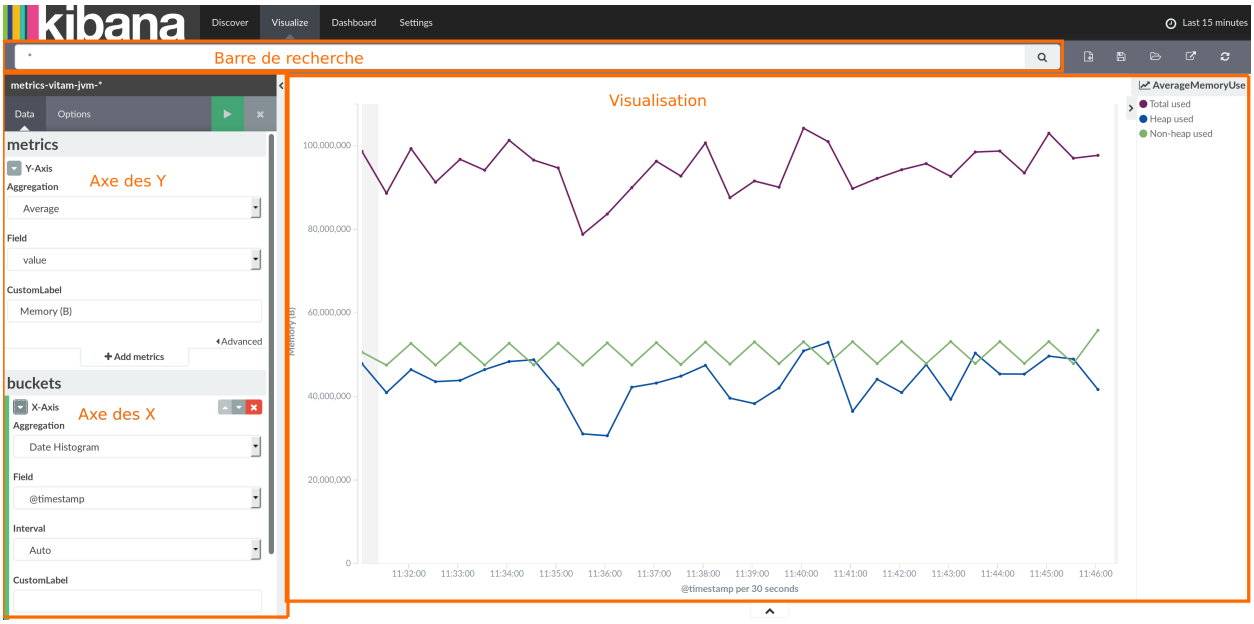
La section **Visualize** permet de consulter les données présentes dans ElasticSearch à travers différents graphiques statistiques. Les graphiques disponibles sont :

https://www.elastic.co/fr/products/kibana
https://www.elastic.co/guide/en/kibana/current/index.html

- **Area chart** : utile pour un regroupement de séries chronologiques dans lequel le total des séries est plus important que la différence entre plusieurs séries.
- **Data table** : un tableau de données classique.
- **Line chart** : graphique pour des séries temporelles. Très utile pour comparer deux séries entre elles.
- **Markdown widget** : utile pour insérer informations sur un dashboard Kibana.
- **Metric** : représentation d'une aggrégation de données sous la forme d'un seul nombre.
- **Pie chart** : un diagramme circulaire classique.
- **Tile map** : représentation de coordonées géographiques sur une carte.
- **Vertical bar chart** : un histogramme classique.

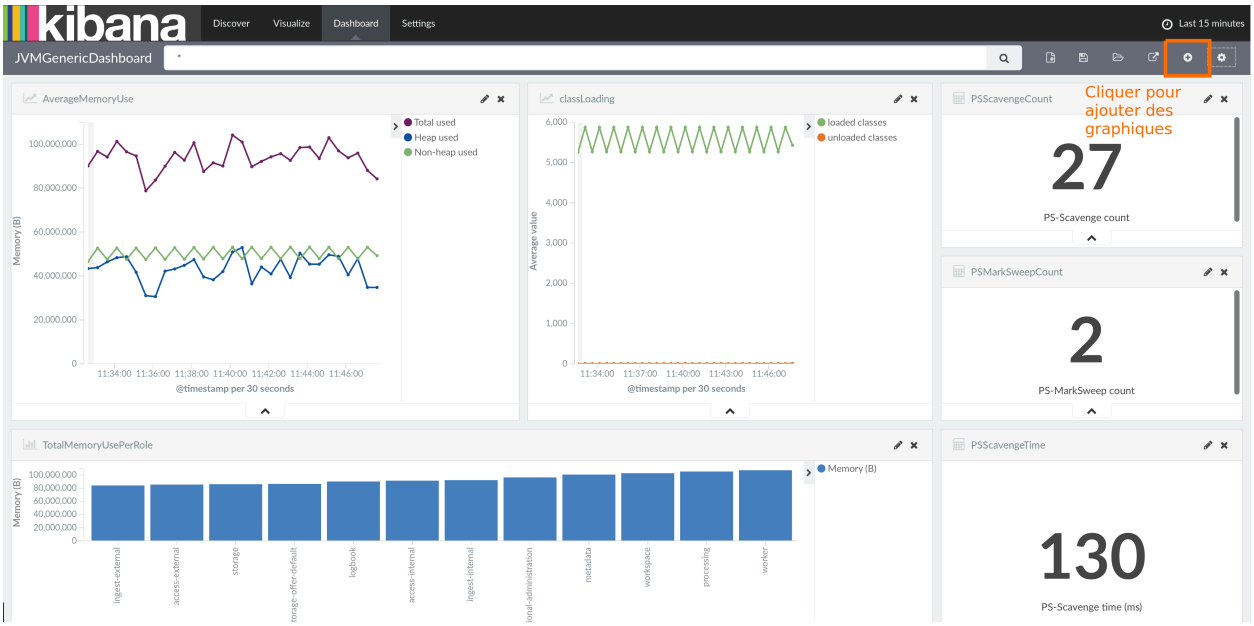
La barre latérale gauche du panneau de visualisation permet de configurer la donnée à représenter. Pour l'axe des Y, il est impératif d'utiliser un aggrégation (moyenne, minimum/maximum, écart type...) sur une valeur pour la représenter. En fonction du graphique sélectionné, il est possible de configurer l'axe des X, toujours au moyen d'aggrégations (dates, date range, terme...).

En haut se situe la même barre de recherche que sur la partie **Discover**, qui permet d'affiner son graphique en effectuant des tris sur sa donnée.



6.2.4.3 Dashboards

La section **Dashboard** permet de regrouper plusieurs graphiques pour constituer un dashboard. Pour ce faire il suffit d'importer des graphiques avec le bouton « + » en haut à droite.



6.3 API de de supervision

Chaque composant *VITAM* peut dialoguer, selon le paramétrage, via 2 réseaux :

- patte d'administration
- patte de service

Si les partitions ne possèdent qu'une seule interface, les deux « pattes » passent par cette unique interface.

6.3.1 Patte d'administration

La solution logicielle *VITAM* expose en interne de la plate-forme les *API REST* suivantes sur ses composants :

- `/admin/v1/status` : statut simple, renvoyant un statut de fonctionnement incluant des informations techniques sur l'état actuel du composant. Un exemple d'utilisation typique est l'intégration à un outil de supervision ou à un élément actif tiers (ex : load-balancer, . . .) . L'appel doit être peu coûteux.
- `/admin/v1/version` : informations de version, build, commit git ayant servi à builder les différents jar.
- `/admin/v1/autotest` : autotest du composant, lançant un test de présence des différentes resources requises par le composant et renvoyant un statut d'état de ces resources.

6.3.1.1 /admin/v1/status

L'API de status renvoie un fichier JSON contenant les informations suivantes :

```
{
    "serverIdentity": {
        "Name":"vitam-iaas-app-01",
        "Role":"logbook",
    "PlatformId":425367
    },
    "status":true,
    "detail": {
```

(suite sur la page suivante)

(suite de la page précédente)

```
    },
        "componentsVersions": {
        "e2eb99d93a74409b3ebc5224e596953e9b8a178f":18
    }
}
```

Signification des champs :

- **serverIdentity**
    - Name : hostname du serveur hébergeant le composant (type : texte)
    - Role : Nom du composant (type : texte)
    - PlatformId : ID de l'environnement (type : entier)
- status : Statut du composant (OK/KO) (type : booléen)
- detail : vide dans cette version, sera défini ultérieurement
- **componentsVersions**
    - hash de commit git : nombre de jars avec buildés depuis ce hash

6.3.1.2 /admin/v1/version

L'API de version renvoie les informations suivantes :

```
[
    {
        "Scm-tags":"",
        "Scm-commit-id":"e2eb99d93a74409b3ebc5224e596953e9b8a178f",
        "Scm-commit-id-abbrev":"e2eb99d",
        "Maven-version":"0.13.0-SNAPSHOT",
        "Scm-dirty":"false",
        "Scm-commit-time":"2017-01-11T16:38:14+01",
        "Maven-build-timestamp":"2017-01-11T16:06:09Z",
        "Scm-branch":"origin/master_iteration_13",
        "Build-Jdk":"1.8.0_111",
        "Maven-artefactId":"logbook-rest",
        "Maven-groupId":"fr.gouv.vitam"
    },
    {
        "Scm-tags":"",
        "Scm-commit-id":"e2eb99d93a74409b3ebc5224e596953e9b8a178f",
        "Scm-commit-id-abbrev":"e2eb99d",
        "Maven-version":"0.13.0-SNAPSHOT",
        "Scm-dirty":"false",
        "Scm-commit-time":"2017-01-11T16:38:14+01",
        "Maven-build-timestamp":"2017-01-11T16:06:09Z",
        "Scm-branch":"origin/master_iteration_13",
        "Build-Jdk":"1.8.0_111",
        "Maven-artefactId":"logbook-administration",
        "Maven-groupId":"fr.gouv.vitam"
    },
    ...
    ...
    ...
]
```

Signification des champs :

- Scm-tags : en cours de définition
- Scm-commit-id : hash de commit git à partir duquel le composant à été buildé
- Scm-commit-id-abbrev : hash de commit abrégé
- Maven-version : Version indiquée à maven dans le fichier pom.xml
- Scm-dirty : Etat du repo git au moment du build (si présence de fichiers unstaged => dirty)
- Scm-commit-time : Date du commit git
- Maven-build-timestamp : Date du build par maven
- Scm-branch : Nom de la branche git à partir de laquelle le composant a été buildé
- Build-Jdk : Version de la jdk ayant servit à builder le composant
- Maven-artefactId : Nom du composant
- Maven-groupId : namespace du composant

6.3.1.3 /admin/v1/autotest

L'API d'autotest renvoie les informations suivantes :


```
{
    "httpCode":200,
    "code":"000000",
    "context":"logbook",
    "state":"OK",
    "message":"All services are available",
    "description":"All services are available",
    "errors": [
        {
            "httpCode":200,
            "code":"1",
            "context":"LogbookMongoDbAccessImpl",
            "state":"OK",
            "message":"Sub service is available",
            "description":"LogbookMongoDbAccessImpl service is available"
        },
        {
            "httpCode":200,
            "code":"2",
            "context":"logbook",
            "state":"OK",
            "message":"Internal service is available",
            "description":"vitam-iaas-app-01 service is available"
        }
    ]
}
```


Signification des champs :

- **httpCode** : code de retour http
- **code** : en cours de définition ; futur code retour interne VITAM
- **context** : Nom du composant
- **state** : Etat du composant (OK/KO)
- **message** : Message de statut
- **description** : Message de description
- **errors :**

- httpCode : code de retour http
- code : code de retour
- context : nom du composant
- state : Etat du composant
- message : Message sur l'état du composant
- description : Description sur l'état du composant

6.3.2 Patte de service

- `/<composant>/v1/status` : statut simple, renvoyant un statut de fonctionnement incluant des informations techniques sur l'état actuel du composant. Un exemple d'utilisation typique est l'intégration à un outil de supervision ou à un élément actif tiers (ex : load-balancer, ...) . L'appel doit être peu coûteux. Le statut normal HTTP renvoyé est 204.

Avertissement : Les composants **vitam-elastic-kibana-interceptor**, **security-internal**, **library** et les *IHM* ne possèdent pas ce statut.

Note : Pour le composant **security-internal**, le point d'*API* est `/v1/status`.

6.4 Logs

La solution logicielle *VITAM* propose une solution ouverte, au choix de l'exploitant. Ce dernier peut, à l'installation, comme à la mise à jour de la solution logicielle *VITAM*, choisir d'utiliser sa propre solution de « regroupement » des logs ou la solution embarquée dans la solution logicielle *VITAM*.

Dans le cas de la solution embarquée, celle-ci se décompose en :

- `rsyslog` ou `syslog-ng` (choix à l'installation, se référer au *DIN*) déployé sur les machines « applicatives » *VITAM* et les envois applicatifs syslog vers un serveur de centralisation de logs (via `facility local0`)
- un serveur de centralisation de logs, comprenant :
  - un mono-noeud (au minimum, ou multi-noeuds) Elasticsearch
  - un moteur logstash, parsant les messages VITAM
  - un afficheur de rendu/aggrégation de données Kibana dédié

Voir aussi :

Les principes & implémentation du système de gestion de logs inclus dans VITAM sont décrits plus en détail dans le DAT.

6.4.1 Changement des règles de log

- Pour les logs fichiers :
  - Définition : fichier `/vitam/conf/<composant>/logback.xml`
  - Format des logs (`encoder`) : ne doit pas être changé ;
  - La sévérité peut être changée ;

  - Roulement : le roulement des fichiers défini par défaut dépend du temps, avec une taille globale maximale ; il est défini par la politique `TimeBasedRollingPolicy` de l'appender RollingFileAppender [15], avec les paramètres suivants :
    - Nombre total de fichiers conservés : 30 (paramètre `maxHistory`) ;
    - Taille totale des fichiers de logs : 5 Go (paramètre `totalSizeCap`) ;
    - Pattern des fichiers : dans le répertoire de logs de l'application : `<service_id>.%d.log` (`%d` étant remplacé par `yyyy-MM-dd`) (paramètre `fileNamePattern`).
- Pour les logs syslog :
  - Format des logs (`suffixPattern`) : ne doit pas être changé ;
  - La sévérité peut être changée ;
  - Les stacktraces sont exclues de l'envoi à la centralisation des logs (paramètre `throwableExcluded` placé à `false`) ; ce paramètre ne doit pas être changé.
- Pour les logs du garbage collector :
  - Niveau de détail : activation des détails et des timestamps (paramètres JVM `-XX:+PrintGCDetails -XX:+PrintGCApplicationStoppedTime`)
  - Roulement : le roulement des fichiers dépend de la taille des fichiers, avec un nombre de fichiers maximal ; il est défini comme suit :
    - Activation du roulement : (paramètre JVM `-XX:+UseGCLogFileRotation`)
    - Nombre total de fichiers conservés : 10 (paramètre JVM `-XX:NumberOfGCLogFiles=10`)
    - Taille unitaire maximale d'un" fichiers de logs : 10 Mo (paramètre JVM `-XX:GCLogFileSize=10M`)
    - Pattern des fichiers : dans le répertoire de logs de l'application (paramètre `-Xloggc:$LOG_FOLDER/gc.log`) pour le fichier courant ; après roulement, les fichiers sont nommés `gc.log.<n>``(avec ``<n>` le numéro du fichier, sur base 0).
- Pour les logs accès :
  - Définition : fichier `/vitam/conf/<service_id>/logback-access.xml`
  - Format des logs (`encoder`) : ne doit pas être changé ;
  - Roulement : le roulement des fichiers défini par défaut dépend du temps, avec une taille globale maximale ; il est défini par la politique `TimeBasedRollingPolicy` de l'appender RollingFileAppender [16], avec les paramètres suivants :
    - Nombre total de fichiers conservés : 7 (paramètre `maxHistory`) ;
    - Taille totale des fichiers de logs : 14 Go (paramètre `totalSizeCap`) ;
    - Pattern des fichiers : dans le répertoire de logs de l'application : `accesslog-<service_id>.%d.log` (`%d` étant remplacé par `yyyy-MM-dd`) (paramètre `fileNamePattern`).

Prudence : La configuration de la durée de rétention des logs accès et/ou leur externalisation devra être ajustée pour respecter les contraintes légales en vigueur pour le système déployé.

6.4.2 Rétention des index sous elasticsearch-log

Curator est l'outil défini dans *VITAM* pour nettoyer les index du *cluster* elasticsearch de log. Curator a été paramétré avec les informations contenues, durant l'installation, dans le fichier `cots_vars.yml`.

Pour les différents index dans le *cluster* Elasticsearch de log, deux paramètres sont définis pour Curator :

http://logback.qos.ch/manual/appenders.html#RollingFileAppender
http://logback.qos.ch/manual/appenders.html#RollingFileAppender

- durée de fermeture : nombre de jours avant clôture de l'index
- durée de suppression : nombre de jours avant suppression de l'index.

Note : concernant les index « logstash-* », il est recommandé de laisser une durée de rétention de 1 an.

Il est possible de modifier le comportement de curator. Pour ce faire, il faut :

1. modifier le fichier `cots_vars.yml`
2. rejouer le playbook de déploiement, en ajoutant en fin de commande `--tags curator_logs`.

6.5 Audit

Divers audits mis à disposition des utilisateurs et administrateurs par le biais de l'*IHM* de démonstration sont décrits dans le Manuel Utilisateurs.

6.5.1 Audit de cohérence

Note : Il est recommandé de procéder à un audit de cohérence aléatoire dans le cadre d'opérations techniques ciblées, telles qu'une migration de plate-forme et de données.

Pour lancer un audit de cohérence, il faut lancer le *playbook* comme suit

```
ansible-playbook -i <inventaire> ansible-playbok-exploitation/audit_coherence.yml --
↪ask-vault-pass -e "access_contract=<contrat multitenant>"
```

Ou, si un fichier vault-password-file existe

```
ansible-playbook -i <inventaire> ansible-playbok-exploitation/audit_coherence.yml --
↪vault-password-file vault_pass.txt -e "access_contract=<contrat multitenant>"
```

Indication : L'audit est lancé sur tous les *tenants* ; cependant, il est nécessaire de donner le contrat d'accès adapté. Se rapprocher du métier pour cet *id* de contrat. Pour limiter la liste des *tenants*, il faut rajouter un *extra var* à la ligne de commande ansible. Exemple

```
-e vitam_tenant_ids=[0,1]
```

pour limiter aux *tenants* 0 et 1.

6.6 Gestion de la capacité

La gestion de la scalabilité du système dépend de ses usages métier ; le lien entre les usages et les composants *VITAM* sollicités est indiqué dans le *DAT*, avec des dimensionnements de plateforme standard pour différents usages.

Le suivi de la charge sur chaque serveur se fait par les outils standard de l'exploitant.

6.7 Suivi de l'état de sécurité

Une étude est actuellement en cours pour réaliser ce type de suivi.

6.8 Alerting

6.8.1 Système

Le suivi des alertes système est à charge de l'exploitant.

6.8.2 Applicatif

Les logs applicatifs de la solution *VITAM* permettent à l'exploitant de mettre en place un *alerting* adapté à l'usage de son équipe métier et technique. Par défaut, et en guise d'exemple, des dashboards Kibana sont disponibles avec un rassemblement des événements courants de sécurité / erreur (ex. : incohérence règles de gestion, désynchronisation MongoDB / ElasticSearch. . .).

6.9 Suivi des Workflows

La solution logicielle *VITAM* intègre une solution de suivi et de gestion des *Workflows*. Elle permet entre autres de :

- Relancer un Workflow arrêté
- Mettre en pause un Workflow démarré
- Rejouer une étape d'un Workflow
- Annuler un workflow

6.9.1 Suivi

Le suivi peut être réalisé via *IHM*, par des appels *REST* ou par un *playbook* ansible.

6.9.1.1 IHM

Il existe une page dans l'*IHM* de démonstration, permettant d'influer sur les processus en cours. Tous les processus mis en pause, automatiquement (lors d'un FATAL) ou bien manuellement (Mode pas à pas) apparaissent sur cette *IHM*. Il est également possible, à partir de cette *IHM*, de relancer le processus ou bien de rejouer une étape, après action d'exploitation.

6.9.1.2 Appels REST

Il est tout aussi possible d'exécuter ces différentes actions sur l'*API* en direct, via des appels `curl` par exemple sur le composant access external :

- PUT sur le endpoint /operations/GUID avec comme header X-Action :RESUME par exemple.

Pour plus d'information, consulter la documentation des *API* externes.

6.9.1.3 *Playbook* ansible

Lancer le script suivant

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/check_
↪workflow_status.yml --vault-password-file vault_pass.txt
```

Ou si le fichier `vault_pass.txt` n'existe pas

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/check_
↪workflow_status.yml --ask-vault-pass
```

Avertissement : Le *playbook* ansible ne peut être exécuté que dans le cas où une installation a déjà été effectuée, et que la *PKI* n'a pas été rejouée (les certificats présents dans `environments/certs` doivent être ceux mis en place dans *VITAM*).

6.9.2 Cas des *worklows* en FATAL

Un *workflow* se met en pause dès qu'il se retrouve en statut **FATAL**. Plusieurs causes peuvent expliquer un tel état.

6.9.2.1 Plugins et Handlers

Plusieurs problèmes peuvent expliquer qu'un *Handler* ou un *plugin* retourne une erreur « FATAL » et donc provoque la mise en pause du *Worfklow*.

Si le composant workspace est défectueux ou ne répond plus, alors un FATAL pourra être obtenu pour tous les *Handlers* et *plugins*.

Si le composant logbook est défectueux ou ne répond plus, alors un FATAL pourra être obtenu pour les *handlers* suivants :

- CommitLifeCycleActionHandler
- CommitLifeCycleObjectGroupActionHandler
- CommitLifeCycleUnitActionHandler
- ListLifecycleTraceabilityActionHandler
- FinalizeLifecycleTraceabilityActionHandler
- RollBackActionHandler

Si le composant functional-administration est défectueux ou ne répond plus, alors un FATAL pourra être obtenu pour les Handlers suivants :

- CheckArchiveProfileRelationActionHandler
- CheckArchiveProfileActionHandler
- GenerateAuditReportActionHandler
- PrepareAuditActionHandler

Si le composant metadata est défectueux ou ne répond plus, alors un FATAL pourra être obtenu pour les Handlers suivants :

- AccessionRegisterActionHandler
- ListArchiveUnitsActionHandler
- PrepareAuditActionHandler
- ArchiveUnitRulesUpdateActionPlugin

- AuditCheckObjectPlugin
- IndexObjectGroupActionPlugin
- IndexUnitActionPlugin
- RunningIngestsUpdateActionPlugin

Si le composant storage est défectueux ou ne répond plus, alors un FATAL pourra être obtenu pour les Handlers suivants :

- CheckStorageAvailabilityActionHandler
- FinalizeLifecycleTraceabilityActionHandler
- GenerateAuditReportActionHandler
- PrepareTraceabilityCheckProcessActionHandler
- PutBinaryOnWorkspace
- CheckIntegrityObjectPlugin
- CheckExistenceObjectPlugin
- StoreMetaDataObjectGroupActionPlugin
- StoreMetaDataUnitActionPlugin
- StoreObjectActionHandler
- StoreObjectGroupActionPlugin

Si le composant processing est défectueux ou ne répond plus, alors un FATAL pourra être obtenu pour les Handlers suivants :

- ListRunningIngestsActionHandler

Si le composant FormatIdentifier est défectueux et ne répond plus, alors un FATAL pourra être obtenu pour le Handler suivant :

- FormatIdentificationActionPlugin

6.9.2.2 Distributor

Plusieurs cas peuvent provoquer un FATAL au niveau du processing :

- si metadata ou workspace est injoignable
- si un *handler* (ou plugin) inexistant est appelé.
- si le distributeur tente d'appeler une famille de worker inexistante

6.9.2.3 Processing - State Machine

Dans le cas ou le Processing ne parvient pas à enregistrer l'état du workflow sur le workspace, un FATAL est provoqué. Il en va de même si le composant logbook est défectueux.

6.9.3 Redémarrer un processus en cas de pause

6.9.3.1 Trouver la cause

De manière générale, il convient d'identifier le composant (ou les composants) posant problème. Il s'agira majoritairement de mtadata, de logbook, du etorage ou encore du workspace.

A partir du Guid de l'opération mise en pause, il est facilement possible de voir, dans les logs du processing ou des workers quels sont les composants incriminés.

6.9.3.2 Relancer le Workflow

A partir du Guid de l'opération mise en pause et une fois le composant redémarré, il est possible de relancer le workflow.

6.9.3.2.1 Vérifier les inputs

S'assurer à partir du GUID de l'opération que l'on nommera X la présence :

- d'un fichier X.json dans /vitam/data/workspace/process/distributorIndex/
- d'un répertoire X dans `/vitam/data/workspace/` contenant à minima une liste de sous-répertoires (et notamment le *SIP* décompressé dans le sous répertoire `SIP`).

6.9.3.2.2 Rejouer une étape

Depuis l'*IHM*, relancer l'étape précédente en cliquant sur l'icône « Replay ». Via les *API*, il suffit de lancer un appel `curl` sur le composant access external : PUT sur le endpoint /operations/GUID avec comme header X-Action :REPLAY.

Cette action aura pour résultat d'exécuter une deuxième fois l'étape qui a échoué. En sortie de ce replay, le statut du workflow doit passer à OK et l'état à PAUSE.

6.9.3.2.3 Prochaine étape

Depuis l'*IHM*, exécuter l'étape suivante en cliquant sur l'icône « Next ». Via les *API*, il suffit de lancer un appel curl sur le composant « access-external » : PUT sur le endpoint /operations/GUID avec comme header X-Action :NEXT.

Cette action aura pour résultat d'exécuter l'étape suivante. En sortie de ce replay, le statut du workflow doit passer à OK et l'état à PAUSE.

6.9.3.2.4 Finaliser le workflow

Il est possible de poursuivre le workflow jusqu'à son terme.

Depuis l'*IHM*, finaliser le workflow en cliquant sur l'icône « Fast Forward ».

Via les *API*, il suffit de lancer un appel curl sur le composant `access-external` : PUT sur le endpoint /operations/GUID avec comme header X-Action :RESUME.

6.10 Cohérence des journaux

Il existe un outil d'administration utilisable par l'exploitant afin de réaliser un test de cohérence des journaux. Cet outil permet de vérifier que les données enregistrées dans la collection LogbookOperations sont bien en cohérence avec les informations sauvegardées dans les collections LFC.

Actuellement, seuls les *TNR* utilisent le point d'API.

A l'avenir, il sera possible de préciser les modalités dans un fichier json associé, et il sera possible d'utiliser le contrôle de cohérence indépendamment.

6.10.1 Lancement

Pour lancer l'outil de cohérence, il suffit de lancer une requête (`curl`, par exemple) sur le serveur logbook interne (sur la « patte » d'administration) :

- POST sur le endpoint /checklogbook

6.10.2 Résultat

L'outil de cohérence renvoie un code OK, si l'opération s'est bien déroulée. En cas d'erreur interne, un code HTTP 500 sera renvoyé.

Dans le cadre d'un OK, un rapport au format Json sera généré, et sera enregistré sur les offres de stockage.

Le rapport contiendra les informations suivantes :

- checkedEvents : la liste des évènements vérifiés.
- checkErrors : la liste des erreurs constatées.

6.11 Liste des *timers* systemd

Note : Dans les sections suivantes, les éléments de type `<curator.log.metrics.close>` correspondent à des variables de l'inventaire ansible utilisé.

Voir aussi :

La fréquence de la plupart *timers* est modifiable (avec un comportement par défaut) ; se reporter au *DIN* et à *Modifier la fréquence de lancement de certains timers systemD* (page 11) pour plus d'informations.

6.11.1 *Timers* de maintenance des index elasticsearch-log

Ces *timers* gèrent la maintenance des index elasticsearch du cluster elasticsearch-log.

Ces *timers* sont activés sur tous les sites d'un déploiement multi-sites.

6.11.1.1 vitam-curator-metrics-indexes

Maintenance des indexes `metrics-vitam-*` (sur elasticsearch-log) (qui contiennent les métriques remontées par les composants VITAM) :

- Ferme les indexes de plus de `<curator.log.metrics.close>` jours ;
- Supprime les indexes de plus de `<curator.log.metrics.delete>` jours.

Units systemd :

- vitam-curator-metrics-indexes.service
- vitam-curator-metrics-indexes.timer

Exécution :

- Localisation : groupe ansible `[hosts_elasticsearch_log]` (sur toutes les instances du groupe)
- Périodicité : Lancé chaque jour à 00 :30.

6.11.1.2 vitam-curator-close-old-indexes

Fermeture des anciens indexes `logstash-*` (sur elasticsearch-log) de plus de `<curator.log.logstash.close>` jours (ces indexes contiennent les logs remontés par les composants et COTS VITAM).

Units systemd :

- vitam-curator-close-old-indexes.service
- vitam-curator-close-old-indexes.timer

Exécution :

- Localisation : groupe ansible `[hosts_elasticsearch_log]` (sur toutes les instances du groupe)
- Périodicité : Lancé chaque jour à 00 :10.

6.11.1.3 vitam-curator-delete-old-indexes

Suppression des indexes `logstash-*` (sur elasticsearch-log) de plus de `<curator.log.logstash.delete>` jours (ces indexes contiennent les logs remontés par les composants et COTS VITAM).

Units systemd :

- vitam-curator-delete-old-indexes.service
- vitam-curator-delete-old-indexes.timer

Exécution :

- Localisation : groupe ansible `[hosts_elasticsearch_log]` (sur toutes les instances du groupe)
- Périodicité : Lancé chaque jour à 00 :20.

6.11.2 *Timers* de gestion des journaux (preuve systémique)

Ces *timers* gèrent la sécurisation des journaux métier *VITAM*.

Ces *timers* sont activés uniquement sur le site primaire d'un déploiement multi-sites.

6.11.2.1 vitam-storage-log-backup

Backup des journaux d'écriture de storage dans les offres de stockage.

Units systemd :

- vitam-storage-log-backup.service
- vitam-storage-log-backup.timer

Exécution :

- Localisation : groupe ansible `[hosts_storage_engine]` (sur toutes les instances du groupe)
- Périodicité : Lancé toutes les heures à 0 minutes 0 secondes, par défaut.

6.11.2.2 vitam-storage-accesslog-backup

Backup des journaux d'accès de storage dans les offres de stockage.

Units systemd :

- vitam-storage-accesslog-backup.service
- vitam-storage-accesslog-backup.timer

Exécution :

- Localisation : groupe ansible `[hosts_storage_engine]` (sur toutes les instances du groupe)
- Périodicité : Lancé toutes les heures à 0 minutes 0 secondes, par défaut.

6.11.2.3 vitam-storage-log-traceability

Sécurisation des journaux d'écriture de storage.

Units systemd :

- vitam-storage-log-traceability.service
- vitam-storage-log-traceability.timer

Exécution :

- Localisation : groupe ansible `[hosts_storage_engine]` (sur la dernière instance du groupe uniquement)
- Périodicité : Lancé toutes les heures à 10 minutes 0 secondes, par défaut.

6.11.2.4 vitam-traceability-operations

Sécurisation du journal des opérations.

Units systemd :

- vitam-traceability-operations.service
- vitam-traceability-operations.timer

Exécution :

- Localisation : groupe ansible `[hosts_logbook]` (sur la dernière instance du groupe uniquement)
- Périodicité : lancé à chaque changement d'heure, par défaut.

6.11.2.5 vitam-traceability-lfc-unit

Sécurisation du journal du cycle de vie des unités archivistiques.

Units systemd :

- vitam-traceability-lfc-unit.service
- vitam-traceability-lfc-unit.timer

Exécution :

- Localisation : groupe ansible `[hosts_logbook]` (sur la dernière instance du groupe uniquement)
- Périodicité : lancé à chaque changement d'heure, par défaut.

6.11.2.6 vitam-traceability-lfc-objectgroup

Sécurisation du journal du cycle de vie des groupes d'objets.

Units systemd :

- vitam-traceability-lfc-objectgroup.service
- vitam-traceability-lfc-objectgroup.timer

Exécution :

- Localisation : groupe ansible `[hosts_logbook]` (sur la dernière instance du groupe uniquement)
- Périodicité : lancé à chaque changement d'heure, par défaut.

6.11.3 *Timers* d'audit interne VITAM

Ces *timers* gèrent le déclenchement périodique des tâches d'audit interne *VITAM*.

Ces *timers* sont activés uniquement sur le site primaire d'un déploiement multi-sites.

6.11.3.1 vitam-traceability-audit

Contrôle de la validité de la sécurisation des journaux.

Units systemd :

- vitam-traceability-audit.service
- vitam-traceability-audit.timer

Exécution :

- Localisation : groupe ansible `[hosts_logbook]` (sur la dernière instance du groupe uniquement)
- Périodicité : lancé chaque jour à 0 :00, par défaut.

6.11.3.2 vitam-rule-management-audit

Validation de la cohérence des règles de gestion entre les offres de stockage et les bases de données.

Units systemd :

- vitam-rule-management-audit.service
- vitam-rule-management-audit.timer

Exécution :

- Localisation : groupe ansible `[hosts_functional_administration]` (sur la dernière instance du groupe uniquement)
- Périodicité : lancé à chaque changement d'heure, par défaut.

6.11.4 *Timer* relatif aux liens symboliques de *accession register*

6.11.4.1 vitam-create-accession-register-symbolic

Déclenche une commande qui va calculer le registre des fonds symbolique et les ajoute dans les bases de données.

Units systemd :

- vitam-create-accession-register-symbolic.service (activé sur site primaire uniquement)

- vitam-create-accession-register-symbolic.timer (activé sur site primaire uniquement)

Exécution :

- Localisation : groupe ansible `[hosts_functional_administration]` (sur la dernière instance du groupe uniquement)
- Périodicité : chaque jour à minuit, par défaut.

6.11.5 *Timers* de reconstruction VITAM

Ces timers gèrent la reconstruction des bases de données VITAM à partir des informations persistées dans les offres de stockage.

Ces timers sont activés uniquement sur le site secondaire d'un déploiement multi-sites.

6.11.5.1 vitam-functional-administration-reconstruction

Reconstruction des données portées par le composant functional-administration.

Units systemd :

- vitam-functional-administration-reconstruction.service
- vitam-functional-administration-reconstruction.timer
- vitam-functional-administration-accession-register-reconstruction.service (activé sur site secondaire seulement)
- vitam-functional-administration-accession-register-reconstruction.timer (activé sur site secondaire seulement)

Exécution :

- Localisation : groupe ansible `[hosts_functional_administration]` (sur la dernière instance du groupe uniquement)
- Périodicité : lancé tous les cinq minutes, par défaut.

6.11.5.2 vitam-logbook-reconstruction

Reconstruction des données portées par le composant logbook.

Units systemd :

- vitam-logbook-reconstruction.service
- vitam-logbook-reconstruction.timer

Exécution :

- Localisation : groupe ansible `[hosts_logbook]` (sur la dernière instance du groupe uniquement)
- Périodicité : lancé tous les 5 minutes, par défaut.

6.11.5.3 vitam-metadata-reconstruction

Reconstruction des données portées par le composant metadata.

Units systemd :

- vitam-metadata-reconstruction.timer
- vitam-metadata-reconstruction.service

Exécution :

- Localisation : groupe ansible `[hosts_metadata]` (sur la dernière instance du groupe uniquement)
- Périodicité : lancé toutes les 5 minutes, par défaut.

6.11.5.4 vitam-metadata-store-graph

Log shipping des données graphes portées par le composant metadata.

Units systemd :

- vitam-metadata-store-graph.timer
- vitam-metadata-store-graph.service

Exécution :

- Localisation : groupe ansible `[hosts_metadata]` (sur la dernière instance du groupe uniquement)
- Périodicité : lancé toutes les 30 minutes, par défaut.

6.11.5.5 vitam-metadata-computed-inherited-rules

Recalcul des *computedInheritedRules* pour les *units* dont les *computedInheritedRules* sont marquées comme obsolètes.

Units systemd :

- vitam-metadata-computed-inherited-rules.timer
- vitam-metadata-computed-inherited-rules.service

Exécution :

- Localisation : groupe ansible `[hosts_metadata]` (sur la dernière instance du groupe uniquement)
- Périodicité : lancé toutes les nuits, à 2h30, par défaut.

6.11.5.6 vitam-metadata-purge-dip

Nettoyage des exports DIPs expirés.

Units systemd :

- vitam-metadata-purge-dip.timer
- vitam-metadata-purge-dip.service

Exécution :

- Localisation : groupe ansible `[hosts_metadata]` (sur la dernière instance du groupe uniquement)
- Périodicité : lancé toutes les nuits, à 2h20, par défaut.

6.11.5.7 vitam-metadata-purge-transfers-SIP

Nettoyage des exports transfers expirés.

Units systemd :

- vitam-metadata-purge-transfers-SIP.timer
- vitam-metadata-purge-transfers-SIP.service

Exécution :

- Localisation : groupe ansible `[hosts_metadata]` (sur la dernière instance du groupe uniquement)
- Périodicité : lancé toutes les nuits, à 2h20, par défaut.

CHAPITRE 7

Exploitation des COTS de la solution logicielle VITAM

7.1 Généralités

Les composants de la solution logicielle *VITAM* sont déployés par un *playbook* ansible qui :

1. déploie, selon l'inventaire employé, les *packages* nécessaires
2. applique la configuration de chaque composant selon son contexte défini dans l'inventaire

Les composants *VITAM* sont décrits ci-après.

Avertissement : En cas de modification de la configuration, redémarrer le service associé.

7.2 COTS

7.2.1 Cerebro

7.2.1.1 Présentation

Cerebro est un utilitaire de supervision de l'état d'un cluster ElasticSearch.

7.2.1.2 Configuration / fichiers utiles

7.2.1.2.1 Fichier `/vitam/conf/cerebro/application.conf`

```
http.port = {{ cerebro.port }}
http.address = {{ ip_admin }}
# Secret will be used to sign session cookies, CSRF tokens and for other encryption␣
↪utilities.
```

(suite sur la page suivante)

(suite de la page précédente)

```
# It is highly recommended to change this value before running cerebro in production.
secret = "{{ cerebro.secret_key }}"

# Application base path
basePath = "/{{ cerebro.baseuri }}/"

# Defaults to RUNNING_PID at the root directory of the app.
# To avoid creating a PID file set this value to /dev/null
pidfile.path = "/dev/null"

# Rest request history max size per user
rest.history.size = 50 // defaults to 50 if not specified

# Path of local database file
data.path = "{{ vitam_defaults.folder.root_path }}/data/cerebro/cerebro.db"

# Authentication
auth = {
  # Example of LDAP authentication
  #type: ldap
    #settings: {
      #url = "ldap://host:port"
      #base-dn = "ou=active,ou=Employee"
      #method  = "simple"
      #user-domain = "domain.com"
    #}
{% if cerebro.basicauth is defined %}
  # Simple username/password authentication
  type: basic
    settings: {
      username = "{{ cerebro.basicauth.username }}"
      password = "{{ cerebro.basicauth.password }}"
    }
{% else %}
  # Example of simple username/password authentication
  #type: basic
    #settings: {
      #username = "admin"
      #password = "1234"
    #}
{% endif %}
}

# A list of known hosts
hosts = [
{% if groups['hosts_elasticsearch_log']|length > 0 %}
  {
    host = "http://{{ elasticsearch.log.host }}:{{ elasticsearch.log.port_http }}"
    name = "{{ elasticsearch.log.cluster_name }}"
  },
{% endif %}
{% if groups['hosts_elasticsearch_data']|length > 0 %}
  {
    host = "http://{{ elasticsearch.data.host }}:{{ elasticsearch.data.port_http }}"
    name = "{{ elasticsearch.data.cluster_name }}"
  },
{% endif %}
```

(suite sur la page suivante)

(suite de la page précédente)

```
  #{
  #  host = "http://localhost:9200"
  #  name = "Some Cluster"
  #},
  # Example of host with authentication
  #{
  #  host = "http://some-authenticated-host:9200"
  #  name = "Secured Cluster"
  #  auth = {
  #    username = "username"
  #    password = "secret-password"
  #  }
  #}
]
```

7.2.1.3 Opérations

- Démarrage du service

Les commandes suivantes sont à passer sur les différentes machines hébergeant le composant vitam-elasticsearch-cerebro.

En tant qu'utilisateur root : `systemctl start vitam-elasticsearch-cerebro`

- Arrêt du service

Les commandes suivantes sont à passer sur les différentes machines constituant le composant vitam-elasticsearch-cerebro.

En tant qu'utilisateur root : `systemctl stop vitam-elasticsearch-cerebro`

- Sauvegarde du service

N/A

- Supervision du service

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:9000/cerebro`

- Exports

N/A

- gestion de la capacité

N/A

- actions récurrentes

N/A

- cas des batches

N/A

7.2.2 consul

7.2.2.1 Présentation

Consul est un DNS applicatif.

7.2.2.1.1 Cas serveur

Le serveur Consul fédère les agents dans leurs requètes « DNS-like » et permet de rebondir sur un DNS externe, s'il ne permet pas de lui-même, de faire la résolution.

7.2.2.1.2 Cas agent

L'agent Consul annonce aux serveurs les services qu'il permet de porter et *checke* régulièrement l'état de ces services.

7.2.2.2 Configuration / fichiers utiles

Les fichiers de configuration sont gérés par les procédures d'installation ou de mise à niveau de l'environnement *VITAM*. Se référer au *DIN*.

7.2.2.2.1 Cas des applicatifs monitorés par Consul

Pour chaque composant *VITAM* nécessitant une supervision de la part de Consul, un fichier est installé sur l'agent de la machine sous `vitam/conf/consul` et est basé sur ce squelette :

7.2.2.2.1.1 Fichier `/vitam/conf/consul/service-<composant>.json`

```
{
  "service": {
{% if vitam_struct.vitam_component == vitam.storageofferdefault.vitam_component %}
    "name": "{{ offer_conf }}",
{% else %}
    "name": "{{ vitam_struct.vitam_component }}",
{% endif %}
    "address": "{{ ip_service }}",
{% if ip_wan is defined %}
    "advertise_addr_wan": "{{ ip_wan }}",
{% endif %}
    "port": {{ vitam_struct.port_service }},
    "enable_tag_override": false,
    "tags": ["vitam","{{ vitam_struct.vitam_component }}"],
    "checks": [
      {
         "name": "{{ vitam_struct.vitam_component }}: business service check",
{% if vitam_struct.https_enabled==true %}
        "notes": "HTTPS port opened",
        "tcp": "{{ ip_service }}:{{ vitam_struct.port_service }}",
{% else %}
        "notes": "HTTP port opened",
        "tcp": "{{ ip_service }}:{{ vitam_struct.port_service }}",
{% endif %}
        "interval": "{{ vitam_struct.consul_check_business }}s"
      },
      {
        "name": "{{ vitam_struct.vitam_component }} : admin service check",
        "notes": "Status admin : /admin/v1/status",
        "http": "http://{{ ip_admin }}:{{ vitam_struct.port_admin }}/admin/v1/status",
```

(suite sur la page suivante)

(suite de la page précédente)

```
        "interval": "{{ vitam_struct.consul_admin_check }}s"
      }
      {% if (vitam_struct.https_enabled != true) and (vitam_struct.vitam_component != vitam.elastickibanainterceptor.vitam_component)  and (vitam_struct.vitam_component != vitam.security_internal.vitam_component) and (vitam_struct.vitam_component != vitam.ihm_demo.vitam_component) and (vitam_struct.vitam_component != vitam.ihm_recette.vitam_component) and (vitam_struct.vitam_component != vitam.library.vitam_component) %}
      ,{
        "name": "{{ vitam_struct.vitam_component }} : http business service check",
        "notes": "Status business : /{{ vitam_struct.baseuri }}/v1/status",
        "http": "http://{{ ip_service }}:{{ vitam_struct.port_service }}/{{ vitam_struct.baseuri }}/v1/status",
        "interval": "{{ vitam_struct.consul_admin_check }}s"
      }
      {% endif %}
      {% if  (vitam_struct.vitam_component == vitam.security_internal.vitam_component)  %}
      ,{
        "name": "{{ vitam_struct.vitam_component }} : http business service check",
        "notes": "Status business : /status",
        "http": "http://{{ ip_service }}:{{ vitam_struct.port_service }}/status",
        "interval": "{{ vitam_struct.consul_check_business }}s"
      }
      {% endif %}
      {% if (vitam_struct.vitam_component == vitam.worker.vitam_component) or (vitam_struct.vitam_component == vitam.ingestexternal.vitam_component)  %}
      ,{
        "name": "Siegfried check",
        "notes": "Is siegfried running ?",
        "tcp": "localhost:{{ siegfried.port }}",
        "interval": "{{ siegfried.consul_check }}s"
      }
{% endif %}
{% if vitam_struct.antivirus is defined %}
      ,{
        "name": "Antivirus check",
        "notes": "Is {{ vitam_struct.antivirus }} running ?",
        "args": ["{{ vitam_folder_conf }}/scan-{{ vitam_struct.antivirus}}.sh","{{ vitam_folder_conf }}/scan-{{ vitam_struct.antivirus}}.sh"],
        "interval": "30s",
        "timeout": "5s"
      }
{% endif %}
    ]
  }
}
```

7.2.2.3 Opérations

- Démarrage du service

En tant qu'utilisateur root : `systemctl start vitam-consul`

- Arrêt du service

En tant qu'utilisateur root : `systemctl stop vitam-consul`

Avertissement : en cas de redémarrage du cluster serveur consul, il faut procéder à un arret/relance par serveur avant de passer au suivant.

- Sauvegarde du service

Ce service ne nécessite pas de sauvegarde particulière.

- Logs

Les logs applicatifs sont envoyés par rsyslog à la solution de centralisation des logs ; il est néanmoins possible d'en virsionner une représentation par la commande :

```
journalctl --unit vitam-consul
```

- Supervision du service

Consul possède une IHM permettant de superviser l'ensemble des services qu'il couvre.

http(s) ://<adresse> :<port>/ui

- Exports

N/A

- gestion de la capacité

N/A

- actions récurrentes
- cas des batches

N/A

7.2.3 Kibana interceptor

7.2.3.1 Présentation

Le composant est une interface d'accès entre kibana « métier » et le *cluster* Elasticsearch de données métier.

Prudence : Ce composant **N'EST PAS** à installer en environnement de production.

7.2.3.2 Configuration / fichiers utiles

Les fichiers de configuration sont définis sous `/vitam/conf/elastic-kibana-interceptor`.

7.2.3.2.1 Fichier `elastic-kibana-interceptor.conf`

```
jettyConfig: jetty-config.xml

clusterName: {{ vitam_struct.cluster_name }}
elasticsearchNodes:
{% for server in groups['hosts_elasticsearch_data'] %}
- hostName: {{ hostvars[server]['ip_service'] }}
  tcpPort: {{ elasticsearch.data.port_http }}
{% endfor %}
whitelist : ["tenant", "all", "mgt", "min", "max", "nbc", "og", "ops", "opi", "sp",
↪"sps", "uds", "up", "us", "storage", "unitType", "v", "qualifiers"]
```

7.2.3.3 Opérations

- Démarrage du service

Les commandes suivantes sont à passer sur les différentes machines constituant le *cluster* Elasticsearch de données.

En tant qu'utilisateur root : `systemctl start vitam-elastic-kibana-interceptor`

- Arrêt du service

Les commandes suivantes sont à passer sur les différentes machines constituant le *cluster* Elasticsearch de données.

En tant qu'utilisateur root : `systemctl stop vitam-elastic-kibana-interceptor`

- Sauvegarde du service

N/A

- Supervision du service

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port>/`

- Exports

N/A

- gestion de la capacité

N/A

- actions récurrentes

- cas des batches

N/A

- Modification de la liste blanche

Modifier dans le fichier `/vitam/conf/elastic-kibana-interceptor/elastic-kibana-interceptor.conf` le contenu de la directive `whitelist`.

A l'issue, redémarrer le composant.

7.2.4 elasticsearch chaîne de log

7.2.4.1 Présentation

Le composant `vitam-elasticsearch-log` est une instance de la base d'indexation elasticsearch stockant les informations suivantes :

- les logs des applications VITAM ;
- les logs des applications du sous-système de centralisation des logs ;
- les métriques applicatives.

7.2.4.2 Configuration / fichiers utiles

Se reporter au *DIN*, qui configure le *cluster* ElastciSearch de la chaîne de log.

Les fichiers de configuration sont définis sous `/vitam/conf/elasticsearch-log`.

7.2.4.2.1 Fichier `/vitam/conf/elasticsearch-log/log4j2.properties`

```
status = error

# log action execution errors for easier debugging
logger.action.name = org.elasticsearch.action
logger.action.level = {{ composant.action_log_level }}

appender.console.type = Console
appender.console.name = console
appender.console.layout.type = PatternLayout
appender.console.layout.pattern = [%d{ISO8601}][%-5p][%-25c{1.}] %marker%m%n

appender.syslog.type = Syslog
appender.syslog.name = syslog
appender.syslog.appName = {{ composant.cluster_name }}
appender.syslog.facility = {{ vitam_defaults.syslog_facility }}
appender.syslog.host = {{ inventory_hostname }}
appender.syslog.protocol = UDP
appender.syslog.port = 514
appender.syslog.layout.type = PatternLayout
# Note: rsyslog only parse RFC3195-formatted syslog messages by default ; AND, to␣
↪make it work with log4j2, we need to start the layout by the app-name.
# IF we were in 5424, we wouldn't have to do this.
appender.syslog.layout.pattern = {{ composant.cluster_name }}: [%d{ISO8601}][%-5p][%-
↪25c{1.}] %marker%m%n
# appender.syslog.format = RFC5424
# appender.syslog.mdcId = esdata

appender.rolling.type = RollingFile
appender.rolling.name = rolling
appender.rolling.fileName = ${sys:es.logs.base_path}${sys:file.separator}${sys:es.
↪logs.cluster_name}.log
appender.rolling.layout.type = PatternLayout
appender.rolling.layout.pattern = [%d{ISO8601}][%-5p][%-25c{1.}] [%node_name]%marker
↪%.-10000m%n
appender.rolling.filePattern = ${sys:es.logs.base_path}${sys:file.separator}${sys:es.
↪logs.cluster_name}-%d{yyyy-MM-dd}-%i.log.gz
appender.rolling.policies.type = Policies
appender.rolling.policies.time.type = TimeBasedTriggeringPolicy
appender.rolling.policies.time.interval = 1
appender.rolling.policies.time.modulate = true
appender.rolling.policies.size.type = SizeBasedTriggeringPolicy
appender.rolling.policies.size.size = {{ composant.log_appenders.rolling.max_log_file_
↪size }}
appender.rolling.strategy.type = DefaultRolloverStrategy
appender.rolling.strategy.fileIndex = nomax
appender.rolling.strategy.action.type = Delete
appender.rolling.strategy.action.basepath = ${sys:es.logs.base_path}
appender.rolling.strategy.action.condition.type = IfFileName
appender.rolling.strategy.action.condition.glob = ${sys:es.logs.cluster_name}-*
appender.rolling.strategy.action.condition.nested_condition.type =␣
↪IfAccumulatedFileSize
appender.rolling.strategy.action.condition.nested_condition.exceeds = {{ composant.
↪log_appenders.rolling.max_total_log_size }}

rootLogger.level = {{ composant.log_appenders.root.log_level }}
rootLogger.appenderRef.console.ref = console
rootLogger.appenderRef.rolling.ref = rolling
```

(suite sur la page suivante)

(suite de la page précédente)

```
rootLogger.appenderRef.syslog.ref = syslog

appender.deprecation_rolling.type = RollingFile
appender.deprecation_rolling.name = deprecation_rolling
appender.deprecation_rolling.fileName = ${sys:es.logs.base_path}${sys:file.separator}$
↪{sys:es.logs.cluster_name}_deprecation.log
appender.deprecation_rolling.layout.type = PatternLayout
appender.deprecation_rolling.layout.pattern = [%d{ISO8601}][%-5p][%-25c{1.}] [%node_
↪name]%marker %.-10000m%n
appender.deprecation_rolling.filePattern = ${sys:es.logs.base_path}${sys:file.
↪separator}${sys:es.logs.cluster_name}_deprecation-%i.log.gz
appender.deprecation_rolling.policies.type = Policies
appender.deprecation_rolling.policies.time.type = TimeBasedTriggeringPolicy
appender.deprecation_rolling.policies.time.interval = 1
appender.deprecation_rolling.policies.time.modulate = true
appender.deprecation_rolling.policies.size.type = SizeBasedTriggeringPolicy
appender.deprecation_rolling.policies.size.size = {{ composant.log_appenders.
↪deprecation_rolling.max_log_file_size }}
appender.deprecation_rolling.strategy.type = DefaultRolloverStrategy
appender.deprecation_rolling.strategy.fileIndex = nomax
appender.deprecation_rolling.strategy.action.type = Delete
appender.deprecation_rolling.strategy.action.basepath = ${sys:es.logs.base_path}
appender.deprecation_rolling.strategy.action.condition.type = IfFileName
appender.deprecation_rolling.strategy.action.condition.glob = ${sys:es.logs.cluster_
↪name}-*
appender.deprecation_rolling.strategy.action.condition.nested_condition.type =␣
↪IfAccumulatedFileSize
appender.deprecation_rolling.strategy.action.condition.nested_condition.exceeds = {{␣
↪composant.log_appenders.deprecation_rolling.max_total_log_size }}

logger.deprecation.name = org.elasticsearch.deprecation
logger.deprecation.level = {{ composant.log_appenders.deprecation_rolling.log_level }}
logger.deprecation.appenderRef.deprecation_rolling.ref = deprecation_rolling
logger.deprecation.additivity = false

appender.index_search_slowlog_rolling.type = RollingFile
appender.index_search_slowlog_rolling.name = index_search_slowlog_rolling
appender.index_search_slowlog_rolling.fileName = ${sys:es.logs.base_path}${sys:file.
↪separator}${sys:es.logs.cluster_name}_index_search_slowlog.log
appender.index_search_slowlog_rolling.layout.type = PatternLayout
appender.index_search_slowlog_rolling.layout.pattern = [%d{ISO8601}][%-5p][%-25c{1.}]␣
↪[%node_name]%marker %.-10000m%n
appender.index_search_slowlog_rolling.filePattern = ${sys:es.logs.base_path}$
↪{sys:file.separator}${sys:es.logs.cluster_name}_index_search_slowlog-%d{yyyy-MM-dd}.
↪log
appender.index_search_slowlog_rolling.policies.type = Policies
appender.index_search_slowlog_rolling.policies.time.type = TimeBasedTriggeringPolicy
appender.index_search_slowlog_rolling.policies.time.interval = 1
appender.index_search_slowlog_rolling.policies.time.modulate = true
appender.index_search_slowlog_rolling.policies.size.type = SizeBasedTriggeringPolicy
appender.index_search_slowlog_rolling.policies.size.size = {{ composant.log_appenders.
↪index_search_slowlog_rolling.max_log_file_size }}
appender.index_search_slowlog_rolling.strategy.type = DefaultRolloverStrategy
appender.index_search_slowlog_rolling.strategy.fileIndex = nomax
appender.index_search_slowlog_rolling.strategy.action.type = Delete
appender.index_search_slowlog_rolling.strategy.action.basepath = ${sys:es.logs.base_
↪path}
```

(suite sur la page suivante)

(suite de la page précédente)

```
appender.index_search_slowlog_rolling.strategy.action.condition.type = IfFileName
appender.index_search_slowlog_rolling.strategy.action.condition.glob = ${sys:es.logs.
↪cluster_name}-*
appender.index_search_slowlog_rolling.strategy.action.condition.nested_condition.type␣
↪= IfAccumulatedFileSize
appender.index_search_slowlog_rolling.strategy.action.condition.nested_condition.
↪exceeds = {{ composant.log_appenders.index_search_slowlog_rolling.max_total_log_
↪size }}

logger.index_search_slowlog_rolling.name = index.search.slowlog
logger.index_search_slowlog_rolling.level = {{ composant.log_appenders.index_search_
↪slowlog_rolling.log_level }}
logger.index_search_slowlog_rolling.appenderRef.index_search_slowlog_rolling.ref =␣
↪index_search_slowlog_rolling
logger.index_search_slowlog_rolling.additivity = false

appender.index_indexing_slowlog_rolling.type = RollingFile
appender.index_indexing_slowlog_rolling.name = index_indexing_slowlog_rolling
appender.index_indexing_slowlog_rolling.fileName = ${sys:es.logs.base_path}${sys:file.
↪separator}${sys:es.logs.cluster_name}_index_indexing_slowlog.log
appender.index_indexing_slowlog_rolling.layout.type = PatternLayout
appender.index_indexing_slowlog_rolling.layout.pattern = [%d{ISO8601}][%-5p][%-25c{1.}
↪] [%node_name]%marker %.-10000m%n
appender.index_indexing_slowlog_rolling.filePattern = ${sys:es.logs.base_path}$
↪{sys:file.separator}${sys:es.logs.cluster_name}_index_indexing_slowlog-%d{yyyy-MM-
↪dd}.log
appender.index_indexing_slowlog_rolling.policies.type = Policies
appender.index_indexing_slowlog_rolling.policies.time.type = TimeBasedTriggeringPolicy
appender.index_indexing_slowlog_rolling.policies.time.interval = 1
appender.index_indexing_slowlog_rolling.policies.time.modulate = true
appender.index_indexing_slowlog_rolling.policies.size.type = SizeBasedTriggeringPolicy
appender.index_indexing_slowlog_rolling.policies.size.size = {{ composant.log_
↪appenders.index_indexing_slowlog_rolling.max_log_file_size }}
appender.index_indexing_slowlog_rolling.strategy.type = DefaultRolloverStrategy
appender.index_indexing_slowlog_rolling.strategy.fileIndex = nomax
appender.index_indexing_slowlog_rolling.strategy.action.type = Delete
appender.index_indexing_slowlog_rolling.strategy.action.basepath = ${sys:es.logs.base_
↪path}
appender.index_indexing_slowlog_rolling.strategy.action.condition.type = IfFileName
appender.index_indexing_slowlog_rolling.strategy.action.condition.glob = ${sys:es.
↪logs.cluster_name}-*
appender.index_indexing_slowlog_rolling.strategy.action.condition.nested_condition.
↪type = IfAccumulatedFileSize
appender.index_indexing_slowlog_rolling.strategy.action.condition.nested_condition.
↪exceeds = {{ composant.log_appenders.index_indexing_slowlog_rolling.max_total_log_
↪size }}

logger.index_indexing_slowlog.name = index.indexing.slowlog.index
logger.index_indexing_slowlog.level = {{ composant.log_appenders.index_indexing_
↪slowlog_rolling.log_level }}
logger.index_indexing_slowlog.appenderRef.index_indexing_slowlog_rolling.ref = index_
↪indexing_slowlog_rolling
logger.index_indexing_slowlog.additivity = false
```

7.2.4.2.2 Fichier `/vitam/conf/elasticsearch-log/jvm.options`

```
## JVM configuration

################################################################
## IMPORTANT: JVM heap size
################################################################
##
## You should always set the min and max JVM heap
## size to the same value. For example, to set
## the heap to 4 GB, set:
##
## -Xms4g
## -Xmx4g
##
## See https://www.elastic.co/guide/en/elasticsearch/reference/current/heap-size.html
## for more information
##
################################################################

# Xms represents the initial size of total heap space
# Xmx represents the maximum size of total heap space

-Xms{{ elasticsearch_memory }}
-Xmx{{ elasticsearch_memory }}

################################################################
## Expert settings
################################################################
##
## All settings below this section are considered
## expert settings. Don't tamper with them unless
## you understand what you are doing
##
################################################################

## GC configuration
-XX:+UseConcMarkSweepGC
-XX:CMSInitiatingOccupancyFraction=75
-XX:+UseCMSInitiatingOccupancyOnly

## optimizations

# pre-touch memory pages used by the JVM during initialization
-XX:+AlwaysPreTouch

## basic

# force the server VM (remove on 32-bit client JVMs)
-server

# explicitly set the stack size (reduce to 320k on 32-bit client JVMs)
-Xss1m

# set to headless, just in case
-Djava.awt.headless=true
```

(suite sur la page suivante)

(suite de la page précédente)

```
# ensure UTF-8 encoding by default (e.g. filenames)
-Dfile.encoding=UTF-8

# use our provided JNA always versus the system one
-Djna.nosys=true

# use old-style file permissions on JDK9
-Djdk.io.permissionsUseCanonicalPath=true

# flags to configure Netty
-Dio.netty.noUnsafe=true
-Dio.netty.noKeySetOptimization=true
-Dio.netty.recycler.maxCapacityPerThread=0

# log4j 2
-Dlog4j.shutdownHookEnabled=false
-Dlog4j2.disable.jmx=true
-Dlog4j.skipJansi=true

## heap dumps

# generate a heap dump when an allocation from the Java heap fails
# heap dumps are created in the working directory of the JVM
-XX:+HeapDumpOnOutOfMemoryError

# specify an alternative path for heap dumps
# ensure the directory exists and has sufficient space
-XX:HeapDumpPath={{ elasticsearch_log_dir }}

## GC logging

-XX:+UseGCLogFileRotation
-XX:NumberOfGCLogFiles=10
-XX:GCLogFileSize=10M
-XX:+PrintGCDetails
-XX:+PrintGCApplicationStoppedTime

#-XX:+PrintGCDetails
#-XX:+PrintGCTimeStamps
#-XX:+PrintGCDateStamps
#-XX:+PrintClassHistogram
#-XX:+PrintTenuringDistribution
#-XX:+PrintGCApplicationStoppedTime

# log GC status to a file with time stamps
# ensure the directory exists
#-Xloggc:${loggc}

# By default, the GC log file will not rotate.
# By uncommenting the lines below, the GC log file
# will be rotated every 128MB at most 32 times.
#-XX:+UseGCLogFileRotation
#-XX:NumberOfGCLogFiles=32
#-XX:GCLogFileSize=128M

# Elasticsearch 5.0.0 will throw an exception on unquoted field names in JSON.
# If documents were already indexed with unquoted fields in a previous version
```

(suite sur la page suivante)

(suite de la page précédente)

```
# of Elasticsearch, some operations may throw errors.
#
# WARNING: This option will be removed in Elasticsearch 6.0.0 and is provided
# only for migration purposes.
#-Delasticsearch.json.allow_unquoted_field_names=true


-Djna.tmpdir={{ vitam_defaults.folder.root_path }}/tmp/{{ composant.cluster_name }}
```

7.2.4.2.3 Fichier `/vitam/conf/elasticsearch-log/elasticsearch.yml`

```
# ======================== Elasticsearch Configuration =========================
#
# NOTE: Elasticsearch comes with reasonable defaults for most settings.
#       Before you set out to tweak and tune the configuration, make sure you
#       understand what are you trying to accomplish and the consequences.
#
# The primary way of configuring a node is via this file. This template lists
# the most important settings you may want to configure for a production cluster.
#
# Please see the documentation for further information on configuration options:
# <http://www.elastic.co/guide/en/elasticsearch/reference/current/setup-configuration.
↪html>
#
# ---------------------------------- Cluster -----------------------------------
#
# Use a descriptive name for your cluster:
#
cluster.name: {{ composant.cluster_name }}
#
# ------------------------------------ Node ------------------------------------
#
# Use a descriptive name for the node:
#
node.name: {{ inventory_hostname }}
# TODO: Better handling of this as we have to modify wich nodes are requested by␣
↪logstash / kibana
node.master: {{ is_master|default('true') }}
node.data: {{ is_data|default('true') }}
#
# Add custom attributes to the node:
#
# node.rack: r1
#
# ----------------------------------- Paths ------------------------------------
#
# Path to directory where to store the data (separate multiple locations by comma):
#
path.data: {{ elasticsearch_data_dir }}
#
# Path to log files:
#
path.logs: {{ elasticsearch_log_dir }}
#
```

(suite sur la page suivante)

(suite de la page précédente)

```
# ---------------------------------- Memory -----------------------------------
#
# Lock the memory on startup:
# = Disable swapping
bootstrap.memory_lock: true
#
# Make sure that the `ES_HEAP_SIZE` environment variable is set to about half the␣
↪memory
# available on the system and that the owner of the process is allowed to use this␣
↪limit.
#
# Elasticsearch performs poorly when the system is swapping the memory.
#
# ---------------------------------- Network -----------------------------------
#
# Set the bind address to a specific IP (IPv4 or IPv6):
#
# Note : if installing to localhost, notably a docker container, we need to bind␣
↪larger than localhost
{% if inventory_hostname in single_vm_hostnames %}
network.host: 0.0.0.0
http.cors.enabled: true
http.cors.allow-origin: "*"
{% else %}
# KWA TODO: Check it again (ansible_hostname VS inventory_hostname VS ip_service)
network.host: {{ ip_admin }}
{% endif %}
# Set a custom port for HTTP:
#
http.port: {{ composant.port_http }}
#network.port: {{ composant.port_tcp }}
transport.tcp.port: {{ composant.port_tcp }}

# For more information, see the documentation at:
# <http://www.elastic.co/guide/en/elasticsearch/reference/current/modules-network.
↪html>
#
# --------------------------------- Discovery ----------------------------------
#
# Pass an initial list of hosts to perform discovery when new node is started:
# The default list of hosts is ["127.0.0.1", "[::1]"]
#
discovery.zen.ping.unicast.hosts: [ {% for host in groups['hosts_elasticsearch_log']
↪%}"{{ hostvars[host]['ip_admin'] }}"{% if not loop.last %},{% endif %}{% endfor %} ]
#
# Prevent the "split brain" by configuring the majority of nodes (total number of␣
↪nodes / 2 + 1):
#
# discovery.zen.minimum_master_nodes: 3
#
# For more information, see the documentation at:
# <http://www.elastic.co/guide/en/elasticsearch/reference/current/modules-discovery.
↪html>
#
# ---------------------------------- Gateway -----------------------------------
#
# Block initial recovery after a full cluster restart until N nodes are started:
```

(suite sur la page suivante)

(suite de la page précédente)

```
#
# gateway.recover_after_nodes: 3
#
# For more information, see the documentation at:
# <http://www.elastic.co/guide/en/elasticsearch/reference/current/modules-gateway.
↪html>
#
# ---------------------------------- Various -----------------------------------
#
# Disable starting multiple nodes on a single system:
#
# node.max_local_storage_nodes: 1
#
# Require explicit names when deleting indices:
#
action.destructive_requires_name: true

# related to https://www.elastic.co/guide/en/elasticsearch/reference/6.8/modules-
↪fielddata.html
indices.fielddata.cache.size: {{ (((elasticsearch_memory_value|int) * (composant.
↪indices_fielddata_cache_size|float)) | round (0, 'floor')) | int }}{{ elasticsearch_
↪memory_unit }}

# related to https://www.elastic.co/guide/en/elasticsearch/reference/6.8/circuit-
↪breaker.html#fielddata-circuit-breaker
indices.breaker.fielddata.limit: {{ (((elasticsearch_memory_value|int) * (composant.
↪indices_breaker_fielddata_limit|float)) | round (0, 'floor')) | int }}{{␣
↪elasticsearch_memory_unit }}
```

7.2.4.2.4 Fichier `/vitam/conf/elasticsearch-log/sysconfig/elasticsearch`

```
################################
# Elasticsearch
################################

# Elasticsearch home directory
#ES_HOME=/usr/share/elasticsearch

# Elasticsearch configuration directory
ES_PATH_CONF={{ vitam_defaults.folder.root_path }}/conf/{{ composant.cluster_name }}

# Elasticsearch data directory
#DATA_DIR={{ vitam_defaults.folder.root_path }}/data/{{ composant.cluster_name }}

# Elasticsearch logs directory
#LOG_DIR={{ vitam_defaults.folder.root_path }}/log/{{ composant.cluster_name }}

# Elasticsearch PID directory
#PID_DIR=/var/run/{{ composant.cluster_name }}

# Heap size defaults to 256m min, 1g max
# Set ES_HEAP_SIZE to 50% of available RAM, but no more than 31g
#ES_JAVA_OPTS=
```

(suite sur la page suivante)

(suite de la page précédente)

```
################################
# Elasticsearch service
################################

# SysV init.d
#
# The number of seconds to wait before checking if Elasticsearch started successfully␣
↪as a daemon process
ES_STARTUP_SLEEP_TIME=5


# Heap new generation
#ES_HEAP_NEWSIZE=

# Maximum direct memory
#ES_DIRECT_SIZE=

# Additional Java OPTS
ES_JAVA_OPTS="-XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=10 -
↪XX:GCLogFileSize=10M -XX:+PrintGCDetails -XX:+PrintGCApplicationStoppedTime"

# Configure restart on package upgrade (true, every other setting will lead to not␣
↪restarting)
#RESTART_ON_UPGRADE=true

# Path to the GC log file
#ES_GC_LOG_FILE={{ vitam_defaults.folder.root_path }}/log/{{ composant.cluster_name }}
↪/gc.log

ES_TMPDIR={{ vitam_defaults.folder.root_path }}/tmp/{{ composant.cluster_name }}

################################
# Elasticsearch service
################################

# SysV init.d
#
# When executing the init script, this user will be used to run the elasticsearch␣
↪service.
# The default value is 'elasticsearch' and is declared in the init.d file.
# Note that this setting is only used by the init script. If changed, make sure that
# the configured user can read and write into the data, work, plugins and log␣
↪directories.
# For systemd service, the user is usually configured in file /usr/lib/systemd/system/
↪elasticsearch.service

# Note: useless for VITAM, as the startup is managed by systemd
ES_USER={{ vitam_defaults.users.vitamdb }}
ES_GROUP={{ vitam_defaults.users.group }}

# The number of seconds to wait before checking if Elasticsearch started successfully␣
↪as a daemon process
ES_STARTUP_SLEEP_TIME=5

################################
# System properties
################################
```

(suite sur la page suivante)

(suite de la page précédente)

```
# Specifies the maximum file descriptor number that can be opened by this process
# When using Systemd, this setting is ignored and the LimitNOFILE defined in
# /usr/lib/systemd/system/elasticsearch.service takes precedence
#MAX_OPEN_FILES=65536

# The maximum number of bytes of memory that may be locked into RAM
# Set to "unlimited" if you use the 'bootstrap.memory_lock: true' option
# in elasticsearch.yml (ES_HEAP_SIZE  must also be set).
# When using Systemd, the LimitMEMLOCK property must be set
# in /usr/lib/systemd/system/elasticsearch.service
#MAX_LOCKED_MEMORY=unlimited

# Maximum number of VMA (Virtual Memory Areas) a process can own
# When using Systemd, this setting is ignored and the 'vm.max_map_count'
# property is set at boot time in /usr/lib/sysctl.d/elasticsearch.conf
#MAX_MAP_COUNT=262144
```

7.2.4.2.5 Fichier `/usr/lib/tmpfiles.d/elasticsearch-log.conf`

```
d    /var/run/{{ composant.cluster_name }}   0755 {{ vitam_defaults.users.vitamdb }} {
↪{ vitam_defaults.users.group }} - -
```

7.2.4.3 Opérations

- Démarrage du service

Les commandes suivantes sont à passer sur les différentes machines constituant le cluster Elasticsearch.

En tant qu'utilisateur root : `systemctl start vitam-elasticsearch-log`

- Arrêt du service

Les commandes suivantes sont à passer sur les différentes machines constituant le cluster Elasticsearch.

En tant qu'utilisateur root : `systemctl stop vitam-elasticsearch-log`

- Sauvegarde du service

Dans cette version du système, seule une sauvegarde à froid du service est supportée (par la sauvegarde des fichiers de données présents dans `/vitam/data`)

- Supervision du service

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port>/`

- Exports

N/A

- gestion de la capacité

N/A

- Réouverture d'un index fermé

Les index sont fermés par action récurrente de Curator ; il est néanmoins possible de rouvrir un index fermé par la commande suivante :

`curl -XPOST '<adresseIP>:<port>/<index_fermé>/_open'`

Référence [17]

[17] https://www.elastic.co/guide/en/elasticsearch/reference/2.4/indices-open-close.html

- actions récurrentes
- cas des batches

N/A

7.2.5 elasticsearch Vitam

7.2.5.1 Présentation

Le composant `vitam-lasticsearch-data` est une instance de la base d'indexation elasticsearch stockant les informations relatives aux archives hébergées dans *VITAM*. Elle participe dans ce sens à l'indexation et la recherche des données contenues dans MongoDB.

7.2.5.2 Configuration / fichiers utiles

Se reporter au *DIN*, qui configure le *cluster* ElastciSearch de données.

Les fichiers de configuration sont définis sous `/vitam/conf/elasticsearch-data`.

7.2.5.2.1 Fichier `log4j2.properties`

```
status = error

# log action execution errors for easier debugging
logger.action.name = org.elasticsearch.action
logger.action.level = {{ composant.action_log_level }}

appender.console.type = Console
appender.console.name = console
appender.console.layout.type = PatternLayout
appender.console.layout.pattern = [%d{ISO8601}][%-5p][%-25c{1.}] %marker%m%n

appender.syslog.type = Syslog
appender.syslog.name = syslog
appender.syslog.appName = {{ composant.cluster_name }}
appender.syslog.facility = {{ vitam_defaults.syslog_facility }}
appender.syslog.host = {{ inventory_hostname }}
appender.syslog.protocol = UDP
appender.syslog.port = 514
appender.syslog.layout.type = PatternLayout
# Note: rsyslog only parse RFC3195-formatted syslog messages by default ; AND, to␣
↪make it work with log4j2, we need to start the layout by the app-name.
# IF we were in 5424, we wouldn't have to do this.
appender.syslog.layout.pattern = {{ composant.cluster_name }}: [%d{ISO8601}][%-5p][%-
↪25c{1.}] %marker%m%n
# appender.syslog.format = RFC5424
# appender.syslog.mdcId = esdata

appender.rolling.type = RollingFile
appender.rolling.name = rolling
appender.rolling.fileName = ${sys:es.logs.base_path}${sys:file.separator}${sys:es.
↪logs.cluster_name}.log
appender.rolling.layout.type = PatternLayout
```

(suite sur la page suivante)

(suite de la page précédente)

```
appender.rolling.layout.pattern = [%d{ISO8601}][%-5p][%-25c{1.}] [%node_name]%marker
↪%.-10000m%n
appender.rolling.filePattern = ${sys:es.logs.base_path}${sys:file.separator}${sys:es.
↪logs.cluster_name}-%d{yyyy-MM-dd}-%i.log.gz
appender.rolling.policies.type = Policies
appender.rolling.policies.time.type = TimeBasedTriggeringPolicy
appender.rolling.policies.time.interval = 1
appender.rolling.policies.time.modulate = true
appender.rolling.policies.size.type = SizeBasedTriggeringPolicy
appender.rolling.policies.size.size = {{ composant.log_appenders.rolling.max_log_file_
↪size }}
appender.rolling.strategy.type = DefaultRolloverStrategy
appender.rolling.strategy.fileIndex = nomax
appender.rolling.strategy.action.type = Delete
appender.rolling.strategy.action.basepath = ${sys:es.logs.base_path}
appender.rolling.strategy.action.condition.type = IfFileName
appender.rolling.strategy.action.condition.glob = ${sys:es.logs.cluster_name}-*
appender.rolling.strategy.action.condition.nested_condition.type =␣
↪IfAccumulatedFileSize
appender.rolling.strategy.action.condition.nested_condition.exceeds = {{ composant.
↪log_appenders.rolling.max_total_log_size }}

rootLogger.level = {{ composant.log_appenders.root.log_level }}
rootLogger.appenderRef.console.ref = console
rootLogger.appenderRef.rolling.ref = rolling
rootLogger.appenderRef.syslog.ref = syslog

appender.deprecation_rolling.type = RollingFile
appender.deprecation_rolling.name = deprecation_rolling
appender.deprecation_rolling.fileName = ${sys:es.logs.base_path}${sys:file.separator}$
↪{sys:es.logs.cluster_name}_deprecation.log
appender.deprecation_rolling.layout.type = PatternLayout
appender.deprecation_rolling.layout.pattern = [%d{ISO8601}][%-5p][%-25c{1.}] [%node_
↪name]%marker %.-10000m%n
appender.deprecation_rolling.filePattern = ${sys:es.logs.base_path}${sys:file.
↪separator}${sys:es.logs.cluster_name}_deprecation-%i.log.gz
appender.deprecation_rolling.policies.type = Policies
appender.deprecation_rolling.policies.time.type = TimeBasedTriggeringPolicy
appender.deprecation_rolling.policies.time.interval = 1
appender.deprecation_rolling.policies.time.modulate = true
appender.deprecation_rolling.policies.size.type = SizeBasedTriggeringPolicy
appender.deprecation_rolling.policies.size.size = {{ composant.log_appenders.
↪deprecation_rolling.max_log_file_size }}
appender.deprecation_rolling.strategy.type = DefaultRolloverStrategy
appender.deprecation_rolling.strategy.fileIndex = nomax
appender.deprecation_rolling.strategy.action.type = Delete
appender.deprecation_rolling.strategy.action.basepath = ${sys:es.logs.base_path}
appender.deprecation_rolling.strategy.action.condition.type = IfFileName
appender.deprecation_rolling.strategy.action.condition.glob = ${sys:es.logs.cluster_
↪name}-*
appender.deprecation_rolling.strategy.action.condition.nested_condition.type =␣
↪IfAccumulatedFileSize
appender.deprecation_rolling.strategy.action.condition.nested_condition.exceeds = {{␣
↪composant.log_appenders.deprecation_rolling.max_total_log_size }}

logger.deprecation.name = org.elasticsearch.deprecation
logger.deprecation.level = {{ composant.log_appenders.deprecation_rolling.log_level }}
```

(suite sur la page suivante)

(suite de la page précédente)

```
logger.deprecation.appenderRef.deprecation_rolling.ref = deprecation_rolling
logger.deprecation.additivity = false

appender.index_search_slowlog_rolling.type = RollingFile
appender.index_search_slowlog_rolling.name = index_search_slowlog_rolling
appender.index_search_slowlog_rolling.fileName = ${sys:es.logs.base_path}${sys:file.
↪separator}${sys:es.logs.cluster_name}_index_search_slowlog.log
appender.index_search_slowlog_rolling.layout.type = PatternLayout
appender.index_search_slowlog_rolling.layout.pattern = [%d{ISO8601}][%-5p][%-25c{1.}]␣
↪[%node_name]%marker %.-10000m%n
appender.index_search_slowlog_rolling.filePattern = ${sys:es.logs.base_path}$
↪{sys:file.separator}${sys:es.logs.cluster_name}_index_search_slowlog-%d{yyyy-MM-dd}.
↪log
appender.index_search_slowlog_rolling.policies.type = Policies
appender.index_search_slowlog_rolling.policies.time.type = TimeBasedTriggeringPolicy
appender.index_search_slowlog_rolling.policies.time.interval = 1
appender.index_search_slowlog_rolling.policies.time.modulate = true
appender.index_search_slowlog_rolling.policies.size.type = SizeBasedTriggeringPolicy
appender.index_search_slowlog_rolling.policies.size.size = {{ composant.log_appenders.
↪index_search_slowlog_rolling.max_log_file_size }}
appender.index_search_slowlog_rolling.strategy.type = DefaultRolloverStrategy
appender.index_search_slowlog_rolling.strategy.fileIndex = nomax
appender.index_search_slowlog_rolling.strategy.action.type = Delete
appender.index_search_slowlog_rolling.strategy.action.basepath = ${sys:es.logs.base_
↪path}
appender.index_search_slowlog_rolling.strategy.action.condition.type = IfFileName
appender.index_search_slowlog_rolling.strategy.action.condition.glob = ${sys:es.logs.
↪cluster_name}-*
appender.index_search_slowlog_rolling.strategy.action.condition.nested_condition.type␣
↪= IfAccumulatedFileSize
appender.index_search_slowlog_rolling.strategy.action.condition.nested_condition.
↪exceeds = {{ composant.log_appenders.index_search_slowlog_rolling.max_total_log_
↪size }}

logger.index_search_slowlog_rolling.name = index.search.slowlog
logger.index_search_slowlog_rolling.level = {{ composant.log_appenders.index_search_
↪slowlog_rolling.log_level }}
logger.index_search_slowlog_rolling.appenderRef.index_search_slowlog_rolling.ref =␣
↪index_search_slowlog_rolling
logger.index_search_slowlog_rolling.additivity = false

appender.index_indexing_slowlog_rolling.type = RollingFile
appender.index_indexing_slowlog_rolling.name = index_indexing_slowlog_rolling
appender.index_indexing_slowlog_rolling.fileName = ${sys:es.logs.base_path}${sys:file.
↪separator}${sys:es.logs.cluster_name}_index_indexing_slowlog.log
appender.index_indexing_slowlog_rolling.layout.type = PatternLayout
appender.index_indexing_slowlog_rolling.layout.pattern = [%d{ISO8601}][%-5p][%-25c{1.}
↪] [%node_name]%marker %.-10000m%n
appender.index_indexing_slowlog_rolling.filePattern = ${sys:es.logs.base_path}$
↪{sys:file.separator}${sys:es.logs.cluster_name}_index_indexing_slowlog-%d{yyyy-MM-
↪dd}.log
appender.index_indexing_slowlog_rolling.policies.type = Policies
appender.index_indexing_slowlog_rolling.policies.time.type = TimeBasedTriggeringPolicy
appender.index_indexing_slowlog_rolling.policies.time.interval = 1
appender.index_indexing_slowlog_rolling.policies.time.modulate = true
appender.index_indexing_slowlog_rolling.policies.size.type = SizeBasedTriggeringPolicy
appender.index_indexing_slowlog_rolling.policies.size.size = {{ composant.log_
↪appenders.index_indexing_slowlog_rolling.max_log_file_size }}
```

(suite sur la page suivante)

(suite de la page précédente)

```
appender.index_indexing_slowlog_rolling.strategy.type = DefaultRolloverStrategy
appender.index_indexing_slowlog_rolling.strategy.fileIndex = nomax
appender.index_indexing_slowlog_rolling.strategy.action.type = Delete
appender.index_indexing_slowlog_rolling.strategy.action.basepath = ${sys:es.logs.base_
↪path}
appender.index_indexing_slowlog_rolling.strategy.action.condition.type = IfFileName
appender.index_indexing_slowlog_rolling.strategy.action.condition.glob = ${sys:es.
↪logs.cluster_name}-*
appender.index_indexing_slowlog_rolling.strategy.action.condition.nested_condition.
↪type = IfAccumulatedFileSize
appender.index_indexing_slowlog_rolling.strategy.action.condition.nested_condition.
↪exceeds = {{ composant.log_appenders.index_indexing_slowlog_rolling.max_total_log_
↪size }}

logger.index_indexing_slowlog.name = index.indexing.slowlog.index
logger.index_indexing_slowlog.level = {{ composant.log_appenders.index_indexing_
↪slowlog_rolling.log_level }}
logger.index_indexing_slowlog.appenderRef.index_indexing_slowlog_rolling.ref = index_
↪indexing_slowlog_rolling
logger.index_indexing_slowlog.additivity = false
```

7.2.5.2.2 Fichier `jvm.options`

```
## JVM configuration

################################################################
## IMPORTANT: JVM heap size
################################################################
##
## You should always set the min and max JVM heap
## size to the same value. For example, to set
## the heap to 4 GB, set:
##
## -Xms4g
## -Xmx4g
##
## See https://www.elastic.co/guide/en/elasticsearch/reference/current/heap-size.html
## for more information
##
################################################################

# Xms represents the initial size of total heap space
# Xmx represents the maximum size of total heap space

-Xms{{ elasticsearch_memory }}
-Xmx{{ elasticsearch_memory }}

################################################################
## Expert settings
################################################################
##
## All settings below this section are considered
## expert settings. Don't tamper with them unless
## you understand what you are doing
```

(suite sur la page suivante)

(suite de la page précédente)

```
##
################################################################

## GC configuration
-XX:+UseConcMarkSweepGC
-XX:CMSInitiatingOccupancyFraction=75
-XX:+UseCMSInitiatingOccupancyOnly

## optimizations

# pre-touch memory pages used by the JVM during initialization
-XX:+AlwaysPreTouch

## basic

# force the server VM (remove on 32-bit client JVMs)
-server

# explicitly set the stack size (reduce to 320k on 32-bit client JVMs)
-Xss1m

# set to headless, just in case
-Djava.awt.headless=true

# ensure UTF-8 encoding by default (e.g. filenames)
-Dfile.encoding=UTF-8

# use our provided JNA always versus the system one
-Djna.nosys=true

# use old-style file permissions on JDK9
-Djdk.io.permissionsUseCanonicalPath=true

# flags to configure Netty
-Dio.netty.noUnsafe=true
-Dio.netty.noKeySetOptimization=true
-Dio.netty.recycler.maxCapacityPerThread=0

# log4j 2
-Dlog4j.shutdownHookEnabled=false
-Dlog4j2.disable.jmx=true
-Dlog4j.skipJansi=true

## heap dumps

# generate a heap dump when an allocation from the Java heap fails
# heap dumps are created in the working directory of the JVM
-XX:+HeapDumpOnOutOfMemoryError

# specify an alternative path for heap dumps
# ensure the directory exists and has sufficient space
-XX:HeapDumpPath={{ elasticsearch_log_dir }}

## GC logging

-XX:+UseGCLogFileRotation
-XX:NumberOfGCLogFiles=10
```

(suite sur la page suivante)

(suite de la page précédente)

```
-XX:GCLogFileSize=10M
-XX:+PrintGCDetails
-XX:+PrintGCApplicationStoppedTime

#-XX:+PrintGCDetails
#-XX:+PrintGCTimeStamps
#-XX:+PrintGCDateStamps
#-XX:+PrintClassHistogram
#-XX:+PrintTenuringDistribution
#-XX:+PrintGCApplicationStoppedTime

# log GC status to a file with time stamps
# ensure the directory exists
#-Xloggc:${loggc}

# By default, the GC log file will not rotate.
# By uncommenting the lines below, the GC log file
# will be rotated every 128MB at most 32 times.
#-XX:+UseGCLogFileRotation
#-XX:NumberOfGCLogFiles=32
#-XX:GCLogFileSize=128M

# Elasticsearch 5.0.0 will throw an exception on unquoted field names in JSON.
# If documents were already indexed with unquoted fields in a previous version
# of Elasticsearch, some operations may throw errors.
#
# WARNING: This option will be removed in Elasticsearch 6.0.0 and is provided
# only for migration purposes.
#-Delasticsearch.json.allow_unquoted_field_names=true


-Djna.tmpdir={{ vitam_defaults.folder.root_path }}/tmp/{{ composant.cluster_name }}
```

7.2.5.2.3 Fichier `elasticsearch.yml`

```
# ======================== Elasticsearch Configuration =========================
#
# NOTE: Elasticsearch comes with reasonable defaults for most settings.
#       Before you set out to tweak and tune the configuration, make sure you
#       understand what are you trying to accomplish and the consequences.
#
# The primary way of configuring a node is via this file. This template lists
# the most important settings you may want to configure for a production cluster.
#
# Please see the documentation for further information on configuration options:
# <http://www.elastic.co/guide/en/elasticsearch/reference/current/setup-configuration.
↪html>
#
# ---------------------------------- Cluster -----------------------------------
#
# Use a descriptive name for your cluster:
#
cluster.name: {{ composant.cluster_name }}
#
```

(suite sur la page suivante)

(suite de la page précédente)

```
# ---------------------------------- Node ------------------------------------
#
# Use a descriptive name for the node:
#
node.name: {{ inventory_hostname }}
# TODO: Better handling of this as we have to modify wich nodes are requested by VITAM
node.master: {{ is_master|default('true') }}
node.data: {{ is_data|default('true') }}
#
# Add custom attributes to the node:
#
# node.rack: r1
#
# ---------------------------------- Paths ------------------------------------
#
# Path to directory where to store the data (separate multiple locations by comma):
#
path.data: {{ elasticsearch_data_dir }}
#
# Path to log files:
#
path.logs: {{ elasticsearch_log_dir }}
#
# ---------------------------------- Memory -----------------------------------
#
# Lock the memory on startup:
# = Disable swapping
bootstrap.memory_lock: true
#
# Make sure that the `ES_HEAP_SIZE` environment variable is set to about half the␣
↪memory
# available on the system and that the owner of the process is allowed to use this␣
↪limit.
#
# Elasticsearch performs poorly when the system is swapping the memory.
#
# ---------------------------------- Network -----------------------------------
#
# Set the bind address to a specific IP (IPv4 or IPv6):
#
# Note : if installing to localhost, notably a docker container, we need to bind␣
↪larger than localhost
{% if inventory_hostname in single_vm_hostnames %}
network.host: 0.0.0.0
http.cors.enabled: true
http.cors.allow-origin: "*"
{% else %}
network.host: {{ ip_service }}
{% endif %}
#
# Set a custom port for HTTP:
#
http.port: {{ composant.port_http }}
transport.tcp.port: {{ composant.port_tcp }}
#
# For more information, see the documentation at:
# <http://www.elastic.co/guide/en/elasticsearch/reference/current/modules-network.
↪html>
```

(suite sur la page suivante)

(suite de la page précédente)

```
#
# ---------------------------------- Discovery -----------------------------------
#
# Pass an initial list of hosts to perform discovery when new node is started:
# The default list of hosts is ["127.0.0.1", "[::1]"]
#
discovery.zen.ping.unicast.hosts: [ {% for host in groups['hosts_elasticsearch_data']
↪%}"{{ hostvars[host]['ip_service'] }}"{% if not loop.last %},{% endif %}{% endfor %}
↪ ]
#
# Prevent the "split brain" by configuring the majority of nodes (total number of␣
↪nodes / 2 + 1):
#
discovery.zen.minimum_master_nodes: {{ ((groups['hosts_elasticsearch_data']|length /␣
↪2)+1)| round (0, 'floor')| int }}
#
# For more information, see the documentation at:
# <http://www.elastic.co/guide/en/elasticsearch/reference/current/modules-discovery.
↪html>
#
# ---------------------------------- Gateway -----------------------------------
#
# Block initial recovery after a full cluster restart until N nodes are started:
#
gateway.expected_nodes: {{ (groups['hosts_elasticsearch_data'] | length) }}
gateway.recover_after_nodes: {{ ((groups['hosts_elasticsearch_data']|length / 2)+1)|␣
↪round (0, 'floor')| int }}
#
# For more information, see the documentation at:
# <http://www.elastic.co/guide/en/elasticsearch/reference/current/modules-gateway.
↪html>
#
# ---------------------------------- Various -----------------------------------
#
# Disable starting multiple nodes on a single system:
#
# node.max_local_storage_nodes: 1
#
# Require explicit names when deleting indices:
#
action.destructive_requires_name: true

# For Vitam multiquery
indices.query.bool.max_clause_count: 10000

{% if composant.index_buffer_size_ratio is defined %}
# some perforamnce tuning ; see https://www.elastic.co/guide/en/elasticsearch/
↪reference/6.4/tune-for-indexing-speed.html
# 0.1 may be enough, cots_vars declares {{ composant.index_buffer_size_ratio }} as␣
↪ratio on total memory {{ elasticsearch_memory }}
indices.memory.index_buffer_size: {{ ((elasticsearch_memory_value|int)*(composant.
↪index_buffer_size_ratio|float))|round (0, 'floor')| int }}{{ elasticsearch_memory_
↪unit }}
{% endif %}

# thread_pool configuration
thread_pool:
```

(suite sur la page suivante)

(suite de la page précédente)

```
    index:
        size: {{ (ansible_processor_cores * ansible_processor_threads_per_core) |␣
↪round (0, 'floor') | int }}
        queue_size: 5000
    get:
        size: {{ (ansible_processor_cores * ansible_processor_threads_per_core)|␣
↪round (0, 'floor') | int }}
        queue_size: 5000
    search:
        size: {{ ((ansible_processor_cores * ansible_processor_threads_per_core * 3 /␣
↪2) + 1) | round (0, 'floor') | int }}
        min_queue_size: 1000
        queue_size: 5000
    write:
        size: {{ (ansible_processor_cores * ansible_processor_threads_per_core + 1)|␣
↪round (0, 'floor') | int }}
        queue_size: 5000
    warmer:
        core: 1
        max: {{ ((ansible_processor_cores * ansible_processor_threads_per_core / 2) +␣
↪0.5) | round (0, 'floor') | int }}
        keep_alive: 2m

# Note : the 0.5 in the previous expression is for there is only 1 CPU (else the␣
↪thread pool size would be zero) ! ; Note bis : max 10 threads #
# Note : in ES5 and further : the thread pool "refresh" is of type scaling with a␣
↪keep-alive of 5m and a max of min(10, (# of available processors)/2)

# related to https://www.elastic.co/guide/en/elasticsearch/reference/6.8/modules-
↪fielddata.html
indices.fielddata.cache.size: {{ (((elasticsearch_memory_value|int) * (composant.
↪indices_fielddata_cache_size|float)) | round (0, 'floor')) | int }}{{ elasticsearch_
↪memory_unit }}

# related to https://www.elastic.co/guide/en/elasticsearch/reference/6.8/circuit-
↪breaker.html#fielddata-circuit-breaker
indices.breaker.fielddata.limit: {{ (((elasticsearch_memory_value|int) * (composant.
↪indices_breaker_fielddata_limit|float)) | round (0, 'floor')) | int }}{{␣
↪elasticsearch_memory_unit }}
```

7.2.5.2.4 Fichier `sysconfig/elasticsearch`

```
################################
# Elasticsearch
################################

# Elasticsearch home directory
#ES_HOME=/usr/share/elasticsearch

# Elasticsearch configuration directory
ES_PATH_CONF={{ vitam_defaults.folder.root_path }}/conf/{{ composant.cluster_name }}

# Elasticsearch data directory
#DATA_DIR={{ vitam_defaults.folder.root_path }}/data/{{ composant.cluster_name }}
```

(suite sur la page suivante)

(suite de la page précédente)

```
# Elasticsearch logs directory
#LOG_DIR={{ vitam_defaults.folder.root_path }}/log/{{ composant.cluster_name }}

# Elasticsearch PID directory
#PID_DIR=/var/run/{{ composant.cluster_name }}

# Heap size defaults to 256m min, 1g max
# Set ES_HEAP_SIZE to 50% of available RAM, but no more than 31g
#ES_JAVA_OPTS=

################################
# Elasticsearch service
################################

# SysV init.d
#
# The number of seconds to wait before checking if Elasticsearch started successfully␣
↪as a daemon process
ES_STARTUP_SLEEP_TIME=5


# Heap new generation
#ES_HEAP_NEWSIZE=

# Maximum direct memory
#ES_DIRECT_SIZE=

# Additional Java OPTS
ES_JAVA_OPTS="-XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=10 -
↪XX:GCLogFileSize=10M -XX:+PrintGCDetails -XX:+PrintGCApplicationStoppedTime"

# Configure restart on package upgrade (true, every other setting will lead to not␣
↪restarting)
#RESTART_ON_UPGRADE=true

# Path to the GC log file
#ES_GC_LOG_FILE={{ vitam_defaults.folder.root_path }}/log/{{ composant.cluster_name }}
↪/gc.log

ES_TMPDIR={{ vitam_defaults.folder.root_path }}/tmp/{{ composant.cluster_name }}

################################
# Elasticsearch service
################################

# SysV init.d
#
# When executing the init script, this user will be used to run the elasticsearch␣
↪service.
# The default value is 'elasticsearch' and is declared in the init.d file.
# Note that this setting is only used by the init script. If changed, make sure that
# the configured user can read and write into the data, work, plugins and log␣
↪directories.
# For systemd service, the user is usually configured in file /usr/lib/systemd/system/
↪elasticsearch.service
```

(suite sur la page suivante)

(suite de la page précédente)

```
# Note: useless for VITAM, as the startup is managed by systemd
ES_USER={{ vitam_defaults.users.vitamdb }}
ES_GROUP={{ vitam_defaults.users.group }}

# The number of seconds to wait before checking if Elasticsearch started successfully␣
↪as a daemon process
ES_STARTUP_SLEEP_TIME=5

################################
# System properties
################################

# Specifies the maximum file descriptor number that can be opened by this process
# When using Systemd, this setting is ignored and the LimitNOFILE defined in
# /usr/lib/systemd/system/elasticsearch.service takes precedence
#MAX_OPEN_FILES=65536

# The maximum number of bytes of memory that may be locked into RAM
# Set to "unlimited" if you use the 'bootstrap.memory_lock: true' option
# in elasticsearch.yml (ES_HEAP_SIZE  must also be set).
# When using Systemd, the LimitMEMLOCK property must be set
# in /usr/lib/systemd/system/elasticsearch.service
#MAX_LOCKED_MEMORY=unlimited

# Maximum number of VMA (Virtual Memory Areas) a process can own
# When using Systemd, this setting is ignored and the 'vm.max_map_count'
# property is set at boot time in /usr/lib/sysctl.d/elasticsearch.conf
#MAX_MAP_COUNT=262144
```

7.2.5.2.5 Fichier `/usr/lib/tmpfiles.d/elasticsearch-data.conf`

```
d    /var/run/{{ composant.cluster_name }}   0755 {{ vitam_defaults.users.vitamdb }} {
↪{ vitam_defaults.users.group }} - -
```

7.2.5.3 Opérations

- Démarrage du service

Les commandes suivantes sont à passer sur les différentes machines constituant le cluster Elasticsearch.

En tant qu'utilisateur root : `systemctl start vitam-elasticsearch-data`

- Arrêt du service

Les commandes suivantes sont à passer sur les différentes machines constituant le cluster Elasticsearch.

En tant qu'utilisateur root : `systemctl stop vitam-elasticsearch-data`

- Sauvegarde du service

Dans cette version du système, seule une sauvegarde à froid du service est supportée (par la sauvegarde des fichiers de données présents dans `/vitam/data`)

- Supervision du service

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port>/`

- Exports

N/A

- gestion de la capacité

N/A

- actions récurrentes
- cas des batches

N/A

7.2.6 Kibana

7.2.6.1 Présentation

Kibana est l'outil permettant de représenter de façon agrégée des données stockées dans ElasticSearch.

Afin de forcer la bonne version de Kibana, *VITAM* déploie un installeur-chapeau `vitam-kibana`.

Prudence : le composant kibana ne peut se connecter qu'à un cluster ElasticSearch ; pour superviser les clusters Elasticsearch de données et de log, il convient de définir des machines différentes (pour chaque kibana) durant l'installation.

VITAM injecte des *dashboards* durant l'installation.

7.2.6.2 Configuration / fichiers utiles

Les fichiers de configuration sont gérés par les procédures d'installation ou de mise à niveau de l'environnement *VITAM*. Se référer au *DIN*.

Le *playbook* d'installation effectue des actions de modification du fichier de configuration standard `/etc/kibana/kibana.yml`.

7.2.6.3 Opérations

- Démarrage du service

En tant qu'utilisateur root : `systemctl start kibana`

- Arrêt du service

En tant qu'utilisateur root : `systemctl stop kibana`

- Sauvegarde du service

Ce service ne nécessite pas de sauvegarde particulière.

- Logs

Les logs applicatifs sont envoyés par rsyslog à la solution de centralisation des logs ; il est néanmoins possible d'en versionner une représentation par la commande :

```
journalctl --unit kibana
```

- Supervision du service

Kibana possède une IHM accessible via la « patte » d'administration :

http(s) ://<adresse> :5601/

- Exports

N/A

- gestion de la capacité

N/A

- actions récurrentes
- cas des batches

N/A

7.2.7 log server

7.2.7.1 Présentation

Ce composant représente en réalité l'ensemble des 3 composants suivants :

- Kibana, pour la présentation des dashboards de logs et de métriques ;
- Logstash, pour l'analyse et la centralisation des logs ;
- Curator, pour la maintenance des index elasticsearch de log.

Le présent chapitre ne s'intéressera qu'à logstash.

7.2.7.2 Configuration / fichiers utiles

L'ansiblerie se charge du paramétrage de ces composants.

7.2.7.3 Opérations

- Démarrage du service

En tant qu'utilisateur root :

Pré-requis : le cluster elasticsearch associé est déjà démarré.

```
systemctl start logstash
```

- Arrêt du service

En tant qu'utilisateur root :

```
systemctl stop logstash
```

Post-requis : le cluster elasticsearch-log associé est arrêté.

- Sauvegarde du service

Ce service ne nécessite pas de sauvegarde particulière.

- Supervision du service

N/A

- Exports

N/A

- gestion de la capacité

N/A

- actions récurrentes

batch Curator, actuellement purgeant les données de plus de XX jours (selon ce qui a été défini dans l'inventaire de ansible) dans Elasticsearch de logs.

- cas des batches

Curator

7.2.8 mongoC

7.2.8.1 Présentation

Replicaset mongoDB servant à stocker la configuration MongoDB (clés de sharding, shards, ...) lors de l'utilisation de MondoDB en mode sharding.

7.2.8.2 Configuration / fichiers utiles

7.2.8.2.1 Fichier `mongoc.conf`

```
# mongod.conf

# for documentation of all options, see:
#   http://docs.mongodb.org/manual/reference/configuration-options/


# where to write logging data.
systemLog:
  destination: syslog
  syslogFacility: local0
  logAppend: true


# Where and how to store data.
storage:
  dbPath: {{ mongo_db_path }}
  directoryPerDB: true


# network interfaces
net:
  port: {{ mongodb.mongoc_port }}
  bindIp: {{ ip_service }}{% if groups['hosts_dev_tools'] | length > 0 and ip_service␣
↪!= ip_admin %},{{ ip_admin }}{% endif %}

  unixDomainSocket:
    enabled: true
    pathPrefix: {{ mongo_tmp_path }}
    filePermissions: 0700


# operationProfiling:
replication:
  replSetName: configsvr # name of the replica set
  enableMajorityReadConcern: true
```

(suite sur la page suivante)

(suite de la page précédente)

```
sharding:
  clusterRole: configsvr # role du shard


# ansible managed security conf
```

7.2.8.2.2 Fichier `keyfile`

```
{{ mongodb[mongo_cluster_name].passphrase }}
```

7.2.8.3 Opérations

- Démarrage du service

En tant qu'utilisateur root : `systemctl start vitam-mongoc`

- Arrêt du service

En tant qu'utilisateur root : `systemctl stop vitam-mongoc`

- Sauvegarde du service

Il est recommandé d'effectuer des sauvegardes régulières des données.

Pour cela, la procédure à suivre est :

1. Arrêt du service
2. Lancement d'un backup (à définir)
3. Démarrage du service

- Supervision du service
- Exports
- gestion de la capacité

N/A

- actions récurrentes
- cas des batches

Cas de l'export tous les soirs/matins ?

7.2.9 mongoD

7.2.9.1 Présentation

Replicaset MongoDB stockant les données métier de Vitam.

7.2.9.2 Configuration / fichiers utiles

7.2.9.2.1 Fichier `mongod.conf`

```
# mongod.conf

# for documentation of all options, see:
#   http://docs.mongodb.org/manual/reference/configuration-options/


# where to write logging data.
systemLog:
  destination: syslog
  syslogFacility: local0
  logAppend: true


# Where and how to store data.
storage:
  dbPath: {{ mongo_db_path }}
  directoryPerDB: true

{% if mongod_memory is defined and mongod_memory != '' %}
  wiredTiger:
    engineConfig:
        cacheSizeGB: {{ mongod_memory }}
{% endif %}


# network interfaces
net:
  port: {{ mongodb.mongod_port }}
  bindIp: {{ ip_service }}{% if groups['hosts_dev_tools'] | length > 0 and ip_service␣
↪!= ip_admin %},{{ ip_admin }}{% endif %}

  unixDomainSocket:
    enabled: true
    pathPrefix: {{ mongo_tmp_path }}
    filePermissions: 0700

# operationProfiling:
replication:
  replSetName: shard{{ mongo_shard_id }} # name of the replica set
  enableMajorityReadConcern: true

sharding:
  clusterRole: shardsvr # role du shard


# ansible managed security conf
```

7.2.9.2.2 Fichier `keyfile`

```
{{ mongodb[mongo_cluster_name].passphrase }}
```

7.2.9.3 Opérations

- Démarrage du service

En tant qu'utilisateur root : `systemctl start vitam-mongod`

- Arrêt du service

En tant qu'utilisateur root : `systemctl stop vitam-mongod`

- Sauvegarde du service

Il est recommandé d'effectuer des sauvegardes régulières des données.

Pour cela, la procédure à suivre est :

1. Arrêt du service
2. Lancement d'un backup (à définir)
3. Démarrage du service

- Supervision du service

Via mongo-express ?

- Exports
- gestion de la capacité

N/A

- actions récurrentes
- cas des batches

Cas de l'export tous les soirs/matins ?

7.2.10 mongoS

7.2.10.1 Présentation

Point d'accès frontal à la base de données MongoDB de Vitam. Redirige sur le bon shard en fonction de la clé de sharding positionnée sur la collection.

7.2.10.2 Configuration / fichiers utiles

7.2.10.2.1 Fichier `mongos.conf`

```
# mongod.conf

# for documentation of all options, see:
#   http://docs.mongodb.org/manual/reference/configuration-options/


# where to write logging data.
systemLog:
  destination: syslog
  syslogFacility: local0
  logAppend: true

# network interfaces
net:
```

(suite sur la page suivante)

(suite de la page précédente)

```
  port: {{ mongodb.mongos_port }}
  bindIp: {{ ip_service }}{% if groups['hosts_dev_tools'] | length > 0 and ip_service␣
↪!= ip_admin %},{{ ip_admin }}{% endif %}

  unixDomainSocket:
    enabled: true
    pathPrefix: {{ mongo_tmp_path }}
    filePermissions: 0700

sharding:
  configDB: configsvr/{% for item in mongoc_list %}{{ hostvars[item]['ip_service'] }}:
↪{{ mongodb.mongoc_port }}{% if not loop.last %},{% endif %}{% endfor %}


# ansible managed security conf
```

7.2.10.2.2 Fichier `keyfile`

```
{{ mongodb[mongo_cluster_name].passphrase }}
```

7.2.10.3 Opérations

- Démarrage du service

En tant qu'utilisateur root : `systemctl start vitam-mongos`

- Arrêt du service

En tant qu'utilisateur root : `systemctl stop vitam-mongos`

- Sauvegarde du service

Il est recommandé d'effectuer des sauvegardes régulières des données.

Pour cela, la procédure à suivre est :

1. Arrêt du service
2. Lancement d'un backup (à définir)
3. Démarrage du service

- Supervision du service

Via mongo-express ?

- Exports
- gestion de la capacité

N/A

- actions récurrentes
- cas des batches

Cas de l'export tous les soirs/matins ?

7.2.11 siegfried

7.2.11.1 Présentation

Siegfried est un outil permettant la détection de format d'un fichier.

7.2.11.2 Configuration / fichiers utiles

Les fichiers de configuration sont gérés par les procédures d'installation ou de mise à niveau de l'environnement *VITAM*. Se référer au *DIN*.

7.2.11.3 Opérations

- Démarrage du service

En tant qu'utilisateur root : `systemctl start vitam-siegfried`

- Arrêt du service

En tant qu'utilisateur root : `systemctl stop vitam-siegfried`

Avertissement : ne pas oublier que cela peut perturber le comportement de certains composants Vitam (ingest-external et worker).

- Sauvegarde du service

Ce service ne nécessite pas de sauvegarde particulière.

- Logs

Les logs applicatifs sont envoyés par rsyslog à la solution de centralisation des logs ; il est néanmoins possible d'en visionner une représentation par la commande :

```
journalctl --unit vitam-siegfried
```

- Supervision du service

N/A

- Exports

N/A

- gestion de la capacité

N/A

- actions récurrentes

- cas des batches

N/A

- Montée de version du fichier de signatures

Se reporter à *Montée de version du fichier de signature de Siegfried* (page 26)

CHAPITRE 8

Exploitation des composants de la solution logicielle VITAM

Les sections qui suivent donnent une description plus fine pour l'exploitation des services VITAM.

8.1 Généralités

Les composants de la solution logicielle *VITAM* sont déployés par un *playbook* ansible qui :

1. déploie, selon l'inventaire employé, les *packages* nécessaires
2. applique la configuration de chaque composant selon son contexte défini dans l'inventaire

Les composants *VITAM* sont décrits ci-après.

Avertissement : En cas de modification de la configuration, redémarrer le service associé.

8.2 Composants

8.2.1 Fichiers communs

Les composants de la solution logicielle *VITAM* utilisent un socle de fichiers communs.

8.2.1.1 Fichier `/vitam/conf/<composant>/sysconfig/java_opts`

Ce fichier définit les JVMARGS.

```
#*****************************************************************************
# Copyright French Prime minister Office/SGMAP/DINSIC/Vitam Program (2015-2019)
#
# contact.vitam@culture.gouv.fr
```

(suite sur la page suivante)

(suite de la page précédente)

```
#
# This software is a computer program whose purpose is to implement a digital archiving back-office system managing
# high volumetry securely and efficiently.
#
# This software is governed by the CeCILL 2.1 license under French law and abiding by the rules of distribution of free
# software. You can use, modify and/ or redistribute the software under the terms of the CeCILL 2.1 license as
# circulated by CEA, CNRS and INRIA at the following URL "http://www.cecill.info".
#
# As a counterpart to the access to the source code and rights to copy, modify and redistribute granted by the license,
# users are provided only with a limited warranty and the software's author, the holder of the economic rights, and the
# successive licensors have only limited liability.
#
# In this respect, the user's attention is drawn to the risks associated with loading, using, modifying and/or
# developing or reproducing the software by the user in light of its specific status of free software, that may mean
# that it is complicated to manipulate, and that also therefore means that it is reserved for developers and
# experienced professionals having in-depth computer knowledge. Users are therefore encouraged to load and test the
# software's suitability as regards their requirements in conditions enabling the security of their systems and/or data
# to be ensured and, more generally, to use and operate it in the same conditions as regards security.
#
# The fact that you are presently reading this means that you have had knowledge of the CeCILL 2.1 license and that you
# accept its terms.
#*******************************************************************************
JAVA_OPTS="{{ vitam_struct.jvm_opts.gc | default(gc_opts) }} {{ vitam_struct.jvm_opts.memory | default(memory_opts) }} {{ vitam_struct.jvm_opts.java | default(java_opts) }} -Dorg.owasp.esapi.resources={{ vitam_folder_conf }} -Dlogback.configurationFile={{ vitam_folder_conf }}/logback.xml -Dvitam.config.folder={{ vitam_folder_conf }} -Dvitam.data.folder={{ vitam_folder_data }} -Dvitam.tmp.folder={{ vitam_folder_tmp }} -Dvitam.log.folder={{ vitam_folder_log }} -Djava.security.properties={{ vitam_folder_conf }}/java.security -XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath={{ vitam_folder_log }}{% if vitam_struct.jvm_log %} -XX:+UnlockDiagnosticVMOptions -XX:+LogVMOutput -XX:LogFile={{ vitam_folder_log }}/jvm.log{% endif %} -XX:+UseG1GC"
JAVA_ARGS="{{ vitam_folder_conf }}/{{ vitam_struct.vitam_component }}.conf"
```

8.2.1.2 Fichier `/vitam/conf/<composant>/logback.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
{% if vitam_struct.logback_rolling_policy|lower == "true" %}

        <appender name="FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
                <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
                        <fileNamePattern>{{ vitam_folder_log }}/accesslog-{{ vitam_struct.vitam_component }}.%d{yyyy-MM-dd}.log</fileNamePattern>
```

(suite sur la page suivante)

(suite de la page précédente)

```
                        <maxHistory>{{ vitam_struct.access_retention_days }}</maxHistory>
                        <totalSizeCap>{{ vitam_struct.access_total_size_cap }}</totalSizeCap>
                </rollingPolicy>
{% else %}

        <appender name="FILE" class="ch.qos.logback.core.FileAppender">
                <file>{{ vitam_folder_log }}/accesslog-{{ vitam_struct.vitam_component }}.log</file>
                <append>true</append>
{% endif %}
                <encoder>
                        <pattern>%h %l %u %t "%r" %s %b "%i{Referer}" "%i{User-agent}" %D %i{X-Request-Id} %i{X-Tenant-Id} %i{X-Application-Id}</pattern>
                </encoder>
        </appender>
        <appender-ref ref="FILE" />
</configuration>
```

8.2.1.3 Fichier `/vitam/conf/<composant>/logback-access.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>

        <!-- Send debug messages to System.out -->
        <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
                <!-- By default, encoders are assigned the type ch.qos.logback.classic.encoder.PatternLayoutEncoder -->
                <encoder>
                        <pattern>%d{ISO8601} [[%thread]] [%X{X-Request-Id}]  %-5level %logger - %replace(%caller{1..2}){'Caller\+1         at |\n',''} : %msg %rootException%n</pattern>
                </encoder>
        </appender>

{% if vitam_struct.logback_rolling_policy|lower == "true" %}
        <appender name="FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
                <rollingPolicy class="ch.qos.logback.core.rolling.SizeAndTimeBasedRollingPolicy">
                        <fileNamePattern>{{ vitam_folder_log }}/{{ vitam_struct.vitam_component }}.%d{yyyy-MM-dd}.%i.log</fileNamePattern>
                        <maxFileSize>{{ vitam_struct.logback_max_file_size }}</maxFileSize>
                        <maxHistory>{{ vitam_struct.logback_total_size_cap.file.history_days }}</maxHistory>
                        <totalSizeCap>{{ vitam_struct.logback_total_size_cap.file.totalsize }}</totalSizeCap>
                </rollingPolicy>
{% else %}
        <appender name="FILE" class="ch.qos.logback.core.FileAppender">
                <file>{{ vitam_folder_log }}/{{ vitam_struct.vitam_component }}.log</file>
                <append>true</append>
{% endif %}
```

(suite sur la page suivante)

(suite de la page précédente)

```
                <encoder>
                        <pattern>%d{ISO8601} [[%thread]] [%X{X-Request-Id}] %-5level
↪%logger - %replace(%caller{1..2}){'Caller\+1     at |\n',''} : %msg %rootException%n
↪</pattern>
                </encoder>
        </appender>

{% if vitam_struct.logback_rolling_policy|lower == "true" %}
        <appender name="SECURITY" class="ch.qos.logback.core.rolling.
↪RollingFileAppender">
                <rollingPolicy class="ch.qos.logback.core.rolling.
↪SizeAndTimeBasedRollingPolicy">
                        <fileNamePattern>{{ vitam_folder_log }}/{{ vitam_struct.vitam_
↪component }}_security.%d{yyyy-MM-dd}.%i.log</fileNamePattern>
                        <maxFileSize>{{ vitam_struct.logback_max_file_size }}</
↪maxFileSize>
                        <maxHistory>{{ vitam_struct.logback_total_size_cap.security.
↪history_days }}</maxHistory>
                    <totalSizeCap>{{ vitam_struct.logback_total_size_cap.security.
↪totalsize }}</totalSizeCap>
                </rollingPolicy>
{% else %}
        <appender name="SECURITY" class="ch.qos.logback.core.FileAppender">
                <file>{{ vitam_folder_log }}/{{ vitam_struct.vitam_component }}_
↪security.log</file>
                <append>true</append>
{% endif %}
                <encoder>
                        <pattern>%d{ISO8601} [[%thread]] [%X{X-Request-Id}] %-5level
↪%logger - %replace(%caller{1..2}){'Caller\+1     at |\n',''} : %msg %rootException%n
                        </pattern>
                </encoder>
        </appender>

{% if vitam_struct.vitam_component == 'storage' %}
 {% if vitam_struct.logback_rolling_policy|lower == "true" %}
        <appender name="OFFERSYNC" class="ch.qos.logback.core.rolling.
↪RollingFileAppender">
                <rollingPolicy class="ch.qos.logback.core.rolling.
↪SizeAndTimeBasedRollingPolicy">
                        <fileNamePattern>{{ vitam_folder_log }}/{{ vitam_struct.vitam_
↪component }}_offer_sync.%d{yyyy-MM-dd}.%i.log</fileNamePattern>
                        <maxFileSize>{{ vitam_struct.logback_max_file_size }}</
↪maxFileSize>
                        <maxHistory>{{ vitam_struct.logback_total_size_cap.offersync.
↪history_days }}</maxHistory>
                        <totalSizeCap>{{ vitam_struct.logback_total_size_cap.
↪offersync.totalsize }}</totalSizeCap>
                </rollingPolicy>
  {% else %}
        <appender name="OFFERSYNC" class="ch.qos.logback.core.FileAppender">
                <file>{{ vitam_folder_log }}/{{ vitam_struct.vitam_component }}_offer_
↪sync.log</file>
                <append>true</append>
  {% endif %}
                <encoder>
                        <pattern>%d{ISO8601} [[%thread]] [%X{X-Request-Id}] %-5level
↪%logger - %replace(%caller{1..2}){'Caller\+1     at |\n',''} : %msg %rootException%n
```

(suite sur la page suivante)

(suite de la page précédente)

```
                        </pattern>
                </encoder>
        </appender>
{% endif %}

{% if vitam_struct.vitam_component == 'offer' %}
  {% if vitam_offers[offer_conf]["provider"] == 'tape-library' %}
    {% if vitam_struct.logback_rolling_policy|lower == "true" %}
        <appender name="OFFER_TAPE" class="ch.qos.logback.core.rolling.RollingFileAppender">
                <rollingPolicy class="ch.qos.logback.core.rolling.SizeAndTimeBasedRollingPolicy">
                        <fileNamePattern>{{ vitam_folder_log }}/{{ vitam_struct.vitam_component }}_offer_tape.%d{yyyy-MM-dd}.%i.log</fileNamePattern>
                        <maxFileSize>{{ vitam_struct.logback_max_file_size }}</maxFileSize>
                        <maxHistory>{{ vitam_struct.logback_total_size_cap.offer_tape.history_days }}</maxHistory>
                        <totalSizeCap>{{ vitam_struct.logback_total_size_cap.offer_tape.totalsize }}</totalSizeCap>
                </rollingPolicy>
    {% else %}
        <appender name="OFFER_TAPE" class="ch.qos.logback.core.FileAppender">
                <file>{{ vitam_folder_log }}/{{ vitam_struct.vitam_component }}_offer_tape.log</file>
                <append>true</append>
    {% endif %}
                <encoder>
                        <pattern>%d{ISO8601} [[%thread]] [%X{X-Request-Id}] %-5level %logger - %replace(%caller{1..2}){'Caller\+1     at |\n',''} : %msg %rootException%n
                        </pattern>
                </encoder>
        </appender>
  {% endif %}
{% endif %}

{% if vitam_struct.vitam_component == 'offer' %}
  {% if vitam_offers[offer_conf]["provider"] == 'tape-library' %}
          {% if vitam_struct.logback_rolling_policy|lower == "true" %}
        <appender name="OFFER_TAPE_BACKUP" class="ch.qos.logback.core.rolling.RollingFileAppender">
                <rollingPolicy class="ch.qos.logback.core.rolling.SizeAndTimeBasedRollingPolicy">
                        <fileNamePattern>{{ vitam_folder_log }}/{{ vitam_struct.vitam_component }}_tape_backup.%d{yyyy-MM-dd}.%i.log</fileNamePattern>
                        <maxFileSize>{{ vitam_struct.logback_max_file_size }}</maxFileSize>
                        <maxHistory>{{ vitam_struct.logback_total_size_cap.offer_tape_backup.history_days }}</maxHistory>
                        <totalSizeCap>{{ vitam_struct.logback_total_size_cap.offer_tape_backup.totalsize }}</totalSizeCap>
                </rollingPolicy>
          {% else %}
        <appender name="OFFER_TAPE_BACKUP" class="ch.qos.logback.core.FileAppender">
                <file>{{ vitam_folder_log }}/{{ vitam_struct.vitam_component }}_tape_backup.log</file>
                <append>true</append>
```

(suite sur la page suivante)

(suite de la page précédente)

```
          {% endif %}
                <encoder>
                        <pattern>%d{ISO8601} [[%thread]] [%X{X-Request-Id}] %-5level
↪%logger - %replace(%caller{1..2}){'Caller\+1     at |\n',''} : %msg %rootException%n
                        </pattern>
                </encoder>
        </appender>
  {% endif %}
{% endif %}

        <appender name="SYSLOG" class="ch.qos.logback.classic.net.SyslogAppender">
                <syslogHost>localhost</syslogHost>
                <facility>{{ vitam_defaults.syslog_facility }}</facility>
                <suffixPattern>vitam-{{ vitam_struct.vitam_component }}: %d{ISO8601}␣
↪[[%thread]] [%X{X-Request-Id}] %-5level %logger - %replace(%caller{1..2}){
↪'Caller\+1     at |\n',''} : %msg %rootException%n</suffixPattern>
        </appender>
        <!-- By default, the level of the root level is set to TRACE -->
        <root level="{{ vitam_struct.log_level | default(vitam_defaults.services.log_
↪level) }}">
                <!-- <appender-ref ref="STDOUT" /> -->
                <appender-ref ref="FILE" />
                <appender-ref ref="SYSLOG" />
        </root>

  <logger name="org.eclipse.jetty" level="WARN"/>
  <logger name="fr.gouv.vitam.storage.engine.server.logbook.StorageLogbookMock" level=
↪"INFO"/>
  <logger name="fr.gouv.vitam.metadata.core.graph.StoreGraphService" level="INFO"/>
  <logger name="fr.gouv.vitam.metadata.core.graph.GraphComputeServiceImpl" level="INFO
↪"/>
  <logger name="fr.gouv.vitam.common" level="WARN" />
{% if vitam_defaults.reconstruction.log_level is defined or vitam_struct.
↪reconstruction.log_level is defined %}
  <logger name="fr.gouv.vitam.metadata.core.reconstruction.ReconstructionService"␣
↪level="{{ vitam_struct.reconstruction.log_level |default(vitam_defaults.
↪reconstruction.log_level) }}"/>
  <logger name="fr.gouv.vitam.metadata.core.reconstruction.RestoreBackupService"␣
↪level="{{ vitam_struct.reconstruction.log_level |default(vitam_defaults.
↪reconstruction.log_level) }}"/>
  <logger name="fr.gouv.vitam.logbook.common.server.reconstruction.
↪ReconstructionService" level="{{ vitam_struct.reconstruction.log_level␣
↪|default(vitam_defaults.reconstruction.log_level) }}"/>
  <logger name="fr.gouv.vitam.logbook.common.server.reconstruction.
↪RestoreBackupService" level="{{ vitam_struct.reconstruction.log_level␣
↪|default(vitam_defaults.reconstruction.log_level) }}"/>
  <logger name="fr.gouv.vitam.functional.administration.common.impl.
↪ReconstructionServiceImpl" level="{{ vitam_struct.reconstruction.log_level␣
↪|default(vitam_defaults.reconstruction.log_level) }}"/>
  <logger name="fr.gouv.vitam.functional.administration.common.impl.
↪RestoreBackupServiceImpl" level="{{ vitam_struct.reconstruction.log_level␣
↪|default(vitam_defaults.reconstruction.log_level) }}"/>
{% endif %}
{% if vitam_struct.performance_logger|lower == "true" %}}
  <logger name="fr.gouv.vitam.common.performance.PerformanceLogger" level="DEBUG"␣
↪additivity="false" >
     <appender-ref ref="SYSLOG" />
```

(suite sur la page suivante)

(suite de la page précédente)

```
  </logger>
{% endif %}
  <logger name="fr.gouv.vitam.common.alert.AlertServiceImpl" level="INFO">
          <appender-ref ref="SECURITY" />
  </logger>

{% if vitam_struct.vitam_component == 'storage' %}
  <logger name="fr.gouv.vitam.storage.engine.server.offersynchronization" level="INFO
↪">
       <appender-ref ref="OFFERSYNC" />
  </logger>
{% endif %}

{% if vitam_struct.vitam_component == 'offer' %}
  <logger name="fr.gouv.vitam.storage.offers.tape.process.ProcessExecutor" level="INFO
↪" additivity="false" >
       <appender-ref ref="OFFER_TAPE" />
  </logger>

  <logger name="fr.gouv.vitam.storage.offers.tape.utils.BackupLogInformation" level=
↪"INFO" additivity="false" >
         <appender-ref ref="OFFER_TAPE_BACKUP" />
  </logger>

{% endif %}


{% if vitam_struct.vitam_component == 'metadata' %}
  <logger name="fr.gouv.vitam.metadata.core.migration" level="INFO"/>
{% endif %}

</configuration>
```

8.2.1.4 Fichier `/vitam/conf/<composant>/jetty-config.xml`

```
<?xml version="1.0"?>
<!DOCTYPE Configure PUBLIC "-//Jetty//Configure//EN" "http://www.eclipse.org/jetty/
↪configure_9_0.dtd">

<!-- ============================================================= -->
<!-- Documentation of this file format can be found at:            -->
<!-- http://wiki.eclipse.org/Jetty/Reference/jetty.xml_syntax      -->
<!--                                                               -->
<!-- Additional configuration files are available in $JETTY_HOME/etc -->
<!-- and can be mixed in. See start.ini file for the default       -->
<!-- configuration files.                                          -->
<!--                                                               -->
<!-- For a description of the configuration mechanism, see the     -->
<!-- output of:                                                    -->
<!--   java -jar start.jar -?                                      -->
<!-- ============================================================= -->

<!-- ============================================================= -->
<!-- Configure a Jetty Server instance with an ID "Server"         -->
<!-- Other configuration files may also configure the "Server"     -->
```

(suite sur la page suivante)

(suite de la page précédente)

```
<!-- ID, in which case they are adding configuration to the same     -->
<!-- instance.  If other configuration have a different ID, they     -->
<!-- will create and configure another instance of Jetty.            -->
<!-- Consult the javadoc of o.e.j.server.Server for all              -->
<!-- configuration that may be set here.                             -->
<!-- ============================================================= -->
<Configure id="Server" class="org.eclipse.jetty.server.Server">


    <!-- ============================================================= -->
    <!-- Add shared Scheduler instance                                 -->
    <!-- ============================================================= -->
    <Call name="addBean">
        <Arg>
            <New class="org.eclipse.jetty.util.thread.ScheduledExecutorScheduler"/>
        </Arg>
    </Call>

    <!-- ============================================================= -->
    <!-- Http Configuration.                                           -->
    <!-- This is a common configuration instance used by all           -->
    <!-- connectors that can carry HTTP semantics (HTTP, HTTPS, SPDY)-->
    <!-- It configures the non wire protocol aspects of the HTTP       -->
    <!-- semantic.                                                     -->
    <!--                                                               -->
    <!-- This configuration is only defined here and is used by        -->
    <!-- reference from the jetty-http.xml, jetty-https.xml and        -->
    <!-- jetty-spdy.xml configuration files which instantiate the      -->
    <!-- connectors.                                                   -->
    <!--                                                               -->
    <!-- Consult the javadoc of o.e.j.server.HttpConfiguration         -->
    <!-- for all configuration that may be set here.                   -->
    <!-- ============================================================= -->
    <New id="httpConfig" class="org.eclipse.jetty.server.HttpConfiguration">
        <Set name="secureScheme">http</Set>
        <Set name="securePort">8443</Set>
        <Set name="outputBufferSize">32768</Set>
        <Set name="requestHeaderSize">8192</Set>
        <Set name="responseHeaderSize">8192</Set>
        <Set name="sendServerVersion">false</Set>
        <Set name="sendDateHeader">false</Set>
        <Set name="headerCacheSize">512</Set>

        <!-- Uncomment to enable handling of X-Forwarded- style headers
        <Call name="addCustomizer">
          <Arg><New class="org.eclipse.jetty.server.ForwardedRequestCustomizer"/></
↪Arg>
        </Call>
        -->
    </New>

    <!-- ======================== Original Connector ================================ -
↪->
    <!-- <Call name="addConnector">
↪                                                                  -->
    <!--     <Arg>
↪                                                                      -->
```

(suite sur la page suivante)

(suite de la page précédente)

```
    <!--          <New class="org.eclipse.jetty.server.ServerConnector">
↪                                  -->
    <!--              <Arg name="server"><Ref refid="Server" /></Arg>
↪                                  -->
    <!--              <Arg name="factories">
↪                                                          -->
    <!--                  <Array type="org.eclipse.jetty.server.ConnectionFactory">
↪                 -->
    <!--                      <Item>
↪                                                                  -->
    <!--                          <New class="org.eclipse.jetty.server.
↪HttpConnectionFactory">        -->
    <!--                              <Arg name="config"><Ref refid="httpConfig" /></
↪Arg>        -->
    <!--                          </New>
↪                                                                  -->
    <!--                      </Item>
↪                                                                  -->
    <!--                  </Array>
↪                                                                  -->
    <!--              </Arg>
↪                                                                      -->
    <!--              <Set name="port">{{ vitam_struct.port_service }}</Set>
↪                                      -->
    <!--              <Set name="idleTimeout">
↪                                                      -->
    <!--                  <Property name="http.timeout" default="{{ vitam_defaults.
↪services.port_service_timeout }}"/>        -->
    <!--              </Set>
↪                                                                      -->
    <!--          </New>
↪                                                                          -->
    <!--      </Arg>
↪                                                                          -->
    <!-- </Call>
↪                                                                          -->



    <!-- =============================================================== -->
    <!-- Set the default handler structure for the Server                -->
    <!-- A handler collection is used to pass received requests to       -->
    <!-- both the ContextHandlerCollection, which selects the next       -->
    <!-- handler by context path and virtual host, and the               -->
    <!-- DefaultHandler, which handles any requests not handled by       -->
    <!-- the context handlers.                                           -->
    <!-- Other handlers may be added to the "Handlers" collection,       -->
    <!-- for example the jetty-requestlog.xml file adds the              -->
    <!-- RequestLogHandler after the default handler                     -->
    <!-- =============================================================== -->
    <Set name="handler">
        <New id="Handlers" class="org.eclipse.jetty.server.handler.HandlerCollection">
            <Set name="handlers">
                <Array type="org.eclipse.jetty.server.Handler">
                    <Item>
                        <New id="Contexts" class="org.eclipse.jetty.server.handler.
↪ContextHandlerCollection"/>
```

(suite sur la page suivante)

(suite de la page précédente)

```
                    </Item>
                    <Item>
                        <New id="DefaultHandler" class="org.eclipse.jetty.server.handler.DefaultHandler"/>
                    </Item>
                </Array>
            </Set>
        </New>
    </Set>

    <Set name="RequestLog">
            <New id="RequestLogImpl" class="ch.qos.logback.access.jetty.RequestLogImpl">
                <Set name="fileName">{{ vitam_folder_conf }}/logback-access.xml</Set>
            </New>
    </Set>
    <Ref id="RequestLogImpl">
              <Call name="start"/>
    </Ref>

    <!-- =========================================================== -->
    <!-- extra server options                                        -->
    <!-- =========================================================== -->
    <Set name="stopAtShutdown">true</Set>
    <Set name="stopTimeout">5000</Set>
    <Set name="dumpAfterStart">false</Set>
    <Set name="dumpBeforeStop">false</Set>

{% if vitam_struct.https_enabled==true %}
   <New id="httpsConfig" class="org.eclipse.jetty.server.HttpConfiguration">
        <Set name="sendServerVersion">false</Set>
        <Set name="sendDateHeader">false</Set>
        <Call name="addCustomizer">
            <Arg>
                <New class="org.eclipse.jetty.server.SecureRequestCustomizer" />
            </Arg>
        </Call>
    </New>
        <New id="sslContextFactory" class="org.eclipse.jetty.util.ssl.SslContextFactory$Server">
        <Set name="KeyStorePath">{{ vitam_folder_conf }}/keystore_{{ vitam_struct.vitam_component }}.jks</Set>
        <Set name="KeyStorePassword">{{ password_keystore }}</Set>
        <Set name="KeyManagerPassword">{{ password_manager_keystore }}</Set>
        <Set name="TrustStorePath">{{ vitam_folder_conf }}/truststore_{{ vitam_struct.vitam_component }}.jks</Set>
        <Set name="TrustStorePassword">{{ password_truststore }}</Set>
        <Set name="TrustStoreType">JKS</Set>
        <Set name="NeedClientAuth">false</Set>
        <Set name="WantClientAuth">true</Set>
      <Set name="IncludeCipherSuites">
        <Array type="String">
          <Item>TLS_ECDHE.*</Item>
          <Item>TLS_DHE_RSA.*</Item>
        </Array>
      </Set>
       <Set name="IncludeProtocols">
```

(suite sur la page suivante)

(suite de la page précédente)

```
        <Array type="String">
          <Item>TLSv1.2</Item>
        </Array>
      </Set>
      <Set name="ExcludeCipherSuites">
        <Array type="String">
          <Item>.*NULL.*</Item>
          <Item>.*RC4.*</Item>
          <Item>.*MD5.*</Item>
          <Item>.*DES.*</Item>
          <Item>.*DSS.</Item>
          <Item>TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA</Item>
          <Item>TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA</Item>
          <Item>TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA</Item>
          <Item>TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA</Item>
          <Item>TLS_DHE_RSA_WITH_AES_256_CBC_SHA</Item>
          <Item>TLS_DHE_RSA_WITH_AES_128_CBC_SHA</Item>
        </Array>
      </Set>
        <Set name="UseCipherSuitesOrder">true</Set>
        <Set name="RenegotiationAllowed">true</Set>
    </New>
    <New id="sslConnectionFactory" class="org.eclipse.jetty.server.
↪SslConnectionFactory">
        <Arg name="sslContextFactory">
            <Ref refid="sslContextFactory" />
        </Arg>
        <Arg name="next">http/1.1</Arg>
    </New>
    <New id="businessConnector" class="org.eclipse.jetty.server.ServerConnector">
        <Arg name="server">
            <Ref refid="Server" />
        </Arg>
        <Arg name="factories">
            <Array type="org.eclipse.jetty.server.ConnectionFactory">
                <Item>
                    <Ref refid="sslConnectionFactory" />
                </Item>
                <Item>
                    <New class="org.eclipse.jetty.server.HttpConnectionFactory">
                        <Arg name="config">
                            <Ref refid="httpsConfig" />
                        </Arg>
                    </New>
                </Item>
            </Array>
        </Arg>
        <Set name="host">{{ ip_service }}</Set>
        <Set name="port">
            <SystemProperty name="jetty.port" default="{{ vitam_struct.port_service }}
↪"/>
        </Set>
        <Set name="name">business</Set>
    </New>

{% else %}

```

(suite sur la page suivante)

(suite de la page précédente)

```
    <!-- ============================================================= -->
    <!-- Connector for API business                                    -->
    <!-- Attach all ContextHanlder except Admin                        -->
    <!-- ============================================================= -->

    <New id="businessConnector" class="org.eclipse.jetty.server.ServerConnector">
        <Arg name="server"><Ref refid="Server" /></Arg>
        <Arg name="factories">
            <Array type="org.eclipse.jetty.server.ConnectionFactory">
                <Item>
                    <New class="org.eclipse.jetty.server.HttpConnectionFactory">
                        <Arg name="config"><Ref refid="httpConfig" /></Arg>
                    </New>
                </Item>
            </Array>
        </Arg>
        <Set name="host">{{ ip_service }}</Set>
        <Set name="port">{{ vitam_struct.port_service }}</Set>
        <Set name="name">business</Set>
        <Set name="idleTimeout">
            <Property name="http.timeout" default="{{ vitam_defaults.services.port_
↪service_timeout }}"/>
        </Set>
    </New>

{% endif %}

    <!-- ============================================================= -->
    <!-- Connector for API Admin                                       -->
    <!-- Attach all ContextHanlder                                     -->
    <!-- ============================================================= -->

    <New id="adminConnector" class="org.eclipse.jetty.server.ServerConnector">
        <Arg name="server"><Ref refid="Server" /></Arg>
        <Arg name="factories">
            <Array type="org.eclipse.jetty.server.ConnectionFactory">
                <Item>
                    <New class="org.eclipse.jetty.server.HttpConnectionFactory">
                        <Arg name="config"><Ref refid="httpConfig" /></Arg>
                    </New>
                </Item>
            </Array>
        </Arg>
        <Set name="host">{{ ip_admin }}</Set>
        <Set name="port">{{ vitam_struct.port_admin }}</Set>
        <Set name="name">admin</Set>
        <Set name="idleTimeout">
            <Property name="http.timeout" default="{{ vitam_defaults.services.port_
↪service_timeout }}"/>
        </Set>
    </New>




    <Call name="setConnectors">
        <Arg>
```

(suite sur la page suivante)

(suite de la page précédente)

```
            <Array type="org.eclipse.jetty.server.ServerConnector">
               <Item>
                   <Ref refid="businessConnector" />
               </Item>
               <Item>
                   <Ref refid="adminConnector" />
               </Item>
            </Array>
        </Arg>
    </Call>

</Configure>
```

8.2.1.5 Fichier /vitam/conf/<composant>/logbook-client.conf

Ce fichier permet de configurer l'appel au composant logbook.

```
serverHost: {{ vitam.logbook.host }}
serverPort: {{ vitam.logbook.port_service }}
```

8.2.1.6 Fichier /vitam/conf/<composant>/server-identity.conf

```
identityName: {{ ansible_nodename }}
identityRole: {{ vitam_struct.vitam_component }}
identitySiteId: {{ vitam_site_id }}
```

8.2.1.7 Fichier /vitam/conf/<composant>/antisamy-esapi.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!--
W3C rules retrieved from:
http://www.w3.org/TR/html401/struct/global.html
-->

<!--
Slashdot allowed tags taken from "Reply" page:
<b> <i> <p> <br> <a> <ol> <ul> <li> <dl> <dt> <dd> <em> <strong> <tt> <blockquote>
↪<div> <ecode> <quote>
-->

<anti-samy-rules xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                 xsi:noNamespaceSchemaLocation="antisamy.xsd">

        <directives>
                <directive name="omitXmlDeclaration" value="true"/>
                <directive name="omitDoctypeDeclaration" value="true"/>
                <directive name="maxInputSize" value="2000000"/>
                <directive name="embedStyleSheets" value="false"/>
        </directives>

```

(suite sur la page suivante)

(suite de la page précédente)

```
23
24          <common-regexps>
25
26                  <!--
27                  From W3C:
28                  This attribute assigns a class name or set of class names to an
29                  element. Any number of elements may be assigned the same class
30                  name or names. Multiple class names must be separated by white
31                  space characters.
32                  -->
33
34                  <regexp name="htmlTitle" value="[a-zA-Z0-9\s-_',:\[\]!\./\\\(\)]*"/>
    ↪<!-- force non-empty with a '+' at the end instead of '*' -->
35                  <regexp name="onsiteURL" value="([\w\\/\.\?=&amp;;\#-~]+|\#(\w)+)"/>
36                  <regexp name="offsiteURL" value="(\s)*((ht|f)tp(s?)://|mailto:)[A-Za-
    ↪z0-9]+[~a-zA-Z0-9-_\.@#$%&amp;;:,\?=/\+!]*(\s)*"/>
37
38          </common-regexps>
39
40          <!--
41
42          Tag.name = a, b, div, body, etc.
43          Tag.action = filter: remove tags, but keep content, validate: keep content as␣
    ↪long as it passes rules, remove: remove tag and contents
44          Attribute.name = id, class, href, align, width, etc.
45          Attribute.onInvalid = what to do when the attribute is invalid, e.g., remove␣
    ↪the tag (removeTag), remove the attribute (removeAttribute), filter the tag␣
    ↪(filterTag)
46          Attribute.description = What rules in English you want to tell the users they␣
    ↪can have for this attribute. Include helpful things so they'll be able to tune␣
    ↪their HTML
47
48           -->
49
50          <!--
51          Some attributes are common to all (or most) HTML tags. There aren't many that␣
    ↪qualify for this. You have to make sure there's no
52          collisions between any of these attribute names with attribute names of other␣
    ↪tags that are for different purposes.
53          -->
54
55          <common-attributes>
56
57
58                  <attribute name="lang" description="The 'lang' attribute tells the␣
    ↪browser what language the element's attribute values and content are written in">
59                          <regexp-list>
60                                  <regexp value="[a-zA-Z]{2,20}"/>
61                          </regexp-list>
62                   </attribute>
63
64                   <attribute name="title" description="The 'title' attribute provides␣
    ↪text that shows up in a 'tooltip' when a user hovers their mouse over the element">
65                          <regexp-list>
66                                  <regexp name="htmlTitle"/>
67                          </regexp-list>
68                   </attribute>
```

(suite sur la page suivante)

(suite de la page précédente)

```
69
70                <attribute name="href" onInvalid="filterTag">
71                        <regexp-list>
72                                <regexp name="onsiteURL"/>
73                                <regexp name="offsiteURL"/>
74                        </regexp-list>
75                </attribute>
76
77                <attribute name="align" description="The 'align' attribute of an HTML element is a direction word, like 'left', 'right' or 'center'">
78                        <literal-list>
79                                <literal value="center"/>
80                                <literal value="left"/>
81                                <literal value="right"/>
82                                <literal value="justify"/>
83                                <literal value="char"/>
84                        </literal-list>
85                </attribute>
86
87        </common-attributes>
88
89
90        <!--
91        This requires normal updates as browsers continue to diverge from the W3C and each other. As long as the browser wars continue
92        this is going to continue. I'm not sure war is the right word for what's going on. Doesn't somebody have to win a war after
93        a while?
94         -->
95
96        <global-tag-attributes>
97                <attribute name="title"/>
98                <attribute name="lang"/>
99        </global-tag-attributes>
100
101
102       <tag-rules>
103
104               <!-- Tags related to JavaScript -->
105
106               <tag name="script" action="remove"/>
107               <tag name="noscript" action="remove"/>
108
109               <!-- Frame & related tags -->
110
111               <tag name="iframe" action="remove"/>
112               <tag name="frameset" action="remove"/>
113               <tag name="frame" action="remove"/>
114               <tag name="noframes" action="remove"/>
115
116
117               <!-- All reasonable formatting tags -->
118
119               <tag name="p" action="validate">
120                       <attribute name="align"/>
121               </tag>
122
```

(suite sur la page suivante)

(suite de la page précédente)

```
                <tag name="div" action="validate"/>
                <tag name="i" action="validate"/>
                <tag name="b" action="validate"/>
                <tag name="em" action="validate"/>
                <tag name="blockquote" action="validate"/>
                <tag name="tt" action="validate"/>

                <tag name="br" action="truncate"/>

                <!-- Custom Slashdot tags, though we're trimming the idea of having a possible mismatching end tag with the endtag="" attribute -->

                <tag name="quote" action="validate"/>
                <tag name="ecode" action="validate"/>


                <!-- Anchor and anchor related tags -->

                <tag name="a" action="validate">

                        <attribute name="href" onInvalid="filterTag"/>
                        <attribute name="nohref">
                                <literal-list>
                                        <literal value="nohref"/>
                                        <literal value=""/>
                                </literal-list>
                        </attribute>
                        <attribute name="rel">
                                <literal-list>
                                        <literal value="nofollow"/>
                                </literal-list>
                        </attribute>
                </tag>

                <!-- List tags -->

                <tag name="ul" action="validate"/>
                <tag name="ol" action="validate"/>
                <tag name="li" action="validate"/>

        </tag-rules>



        <!--  No CSS on Slashdot posts -->

        <css-rules>
        </css-rules>


        <html-entities>
                <entity name="amp" cdata="&amp;"/>
                <entity name="nbsp" cdata="&amp;#160;"/>

                <entity name="iexcl" cdata="&amp;#161;"/> <!--inverted exclamation mark, U+00A1 ISOnum -->
                <entity name="cent" cdata="&amp;#162;"/> <!--cent sign, U+00A2 ISOnum -->
```

(suite sur la page suivante)

(suite de la page précédente)

```
178                 <entity name="pound" cdata="&amp;#163;"/> <!--pound sign, U+00A3␣
↪ISOnum -->
179                 <entity name="curren" cdata="&amp;#164;"/> <!--currency sign, U+00A4␣
↪ISOnum -->
180                 <entity name="yen" cdata="&amp;#165;"/> <!--yen sign = yuan sign,␣
↪U+00A5 ISOnum -->
181                 <entity name="brvbar" cdata="&amp;#166;"/> <!--broken bar = broken␣
↪vertical bar, U+00A6 ISOnum -->
182                 <entity name="sect" cdata="&amp;#167;"/> <!--section sign, U+00A7␣
↪ISOnum -->
183                 <entity name="uml" cdata="&amp;#168;"/> <!--diaeresis = spacing␣
↪diaeresis, U+00A8 ISOdia -->
184                 <entity name="copy" cdata="&amp;#169;"/> <!--copyright sign, U+00A9␣
↪ISOnum -->
185                 <entity name="ordf" cdata="&amp;#170;"/> <!--feminine ordinal␣
↪indicator, U+00AA ISOnum -->
186                 <entity name="laquo" cdata="&amp;#171;"/> <!--left-pointing double␣
↪angle quotation mark = left pointing guillemet, U+00AB ISOnum -->
187                 <entity name="not" cdata="&amp;#172;"/> <!--not sign, U+00AC ISOnum --
↪>
188                 <entity name="shy" cdata="&amp;#173;"/> <!--soft hyphen =␣
↪discretionary hyphen,U+00AD ISOnum -->
189                 <entity name="reg" cdata="&amp;#174;"/> <!--registered sign =␣
↪registered trade mark sign, U+00AE ISOnum -->
190                 <entity name="macr" cdata="&amp;#175;"/> <!--macron = spacing macron␣
↪= overline = APL overbar, U+00AF ISOdia -->
191                 <entity name="deg" cdata="&amp;#176;"/> <!--degree sign, U+00B0␣
↪ISOnum -->
192                 <entity name="plusmn" cdata="&amp;#177;"/> <!--plus-minus sign = plus-
↪or-minus sign, U+00B1 ISOnum -->
193                 <entity name="sup2" cdata="&amp;#178;"/> <!--superscript two =␣
↪superscript digit two = squared, U+00B2 ISOnum -->
194                 <entity name="sup3" cdata="&amp;#179;"/> <!--superscript three =␣
↪superscript digit three= cubed, U+00B3 ISOnum -->
195                 <entity name="acute" cdata="&amp;#180;"/> <!--acute accent = spacing␣
↪acute, U+00B4 ISOdia -->
196                 <entity name="micro" cdata="&amp;#181;"/> <!--micro sign, U+00B5␣
↪ISOnum -->
197                 <entity name="para" cdata="&amp;#182;"/> <!--pilcrow sign = paragraph␣
↪sign, U+00B6 ISOnum -->
198                 <entity name="middot" cdata="&amp;#183;"/> <!--middle dot = Georgian␣
↪comma = Greek middle dot, U+00B7 ISOnum -->
199                 <entity name="cedil" cdata="&amp;#184;"/> <!--cedilla = spacing␣
↪cedilla, U+00B8 ISOdia -->
200                 <entity name="sup1" cdata="&amp;#185;"/> <!--superscript one =␣
↪superscript digit one,U+00B9 ISOnum -->
201                 <entity name="ordm" cdata="&amp;#186;"/> <!--masculine ordinal␣
↪indicator, U+00BA ISOnum -->
202                 <entity name="raquo" cdata="&amp;#187;"/> <!--right-pointing double␣
↪angle quotation mark = right pointing guillemet, U+00BB ISOnum -->
203                 <entity name="frac14" cdata="&amp;#188;"/> <!--vulgar fraction one␣
↪quarter = fraction one quarter, U+00BC ISOnum -->
204                 <entity name="frac12" cdata="&amp;#189;"/> <!--vulgar fraction one␣
↪half = fraction one half, U+00BD ISOnum -->
205                 <entity name="frac34" cdata="&amp;#190;"/> <!--vulgar fraction three␣
↪quarters = fraction three quarters, U+00BE ISOnum -->
206                 <entity name="iquest" cdata="&amp;#191;"/> <!--inverted question mark␣
↪= turned question mark, U+00BF ISOnum -->
```

(suite sur la page suivante)

(suite de la page précédente)

```
207                 <entity name="Agrave" cdata="&amp;#192;"/> <!--latin capital letter A␣
↪with grave = latin capital letter A grave,U+00C0 ISOlat1 -->
208                 <entity name="Aacute" cdata="&amp;#193;"/> <!--latin capital letter A␣
↪with acute,U+00C1 ISOlat1 -->
209                 <entity name="Acirc" cdata="&amp;#194;"/> <!--latin capital letter A␣
↪with circumflex,U+00C2 ISOlat1 -->
210                 <entity name="Atilde" cdata="&amp;#195;"/> <!--latin capital letter A␣
↪with tilde,U+00C3 ISOlat1 -->
211                 <entity name="Auml" cdata="&amp;#196;"/> <!--latin capital letter A␣
↪with diaeresis,U+00C4 ISOlat1 -->
212                 <entity name="Aring" cdata="&amp;#197;"/> <!--latin capital letter A␣
↪with ring above = latin capital letter A ring, U+00C5 ISOlat1 -->
213                 <entity name="AElig" cdata="&amp;#198;"/> <!--latin capital letter AE␣
↪= latin capital ligature AE, U+00C6 ISOlat1 -->
214                 <entity name="Ccedil" cdata="&amp;#199;"/> <!--latin capital letter C␣
↪with cedilla, U+00C7 ISOlat1 -->
215                 <entity name="Egrave" cdata="&amp;#200;"/> <!--latin capital letter E␣
↪with grave, U+00C8 ISOlat1 -->
216                 <entity name="Eacute" cdata="&amp;#201;"/> <!--latin capital letter E␣
↪with acute,U+00C9 ISOlat1 -->
217                 <entity name="Ecirc" cdata="&amp;#202;"/> <!--latin capital letter E␣
↪with circumflex,U+00CA ISOlat1 -->
218                 <entity name="Euml" cdata="&amp;#203;"/> <!--latin capital letter E␣
↪with diaeresis, U+00CB ISOlat1 -->
219                 <entity name="Igrave" cdata="&amp;#204;"/> <!--latin capital letter I␣
↪with grave, U+00CC ISOlat1 -->
220                 <entity name="Iacute" cdata="&amp;#205;"/> <!--latin capital letter I␣
↪with acute, U+00CD ISOlat1 -->
221                 <entity name="Icirc" cdata="&amp;#206;"/> <!--latin capital letter I␣
↪with circumflex, U+00CE ISOlat1 -->
222                 <entity name="Iuml" cdata="&amp;#207;"/> <!--latin capital letter I␣
↪with diaeresis, U+00CF ISOlat1 -->
223                 <entity name="ETH" cdata="&amp;#208;"/> <!--latin capital letter ETH,␣
↪U+00D0 ISOlat1 -->
224                 <entity name="Ntilde" cdata="&amp;#209;"/> <!--latin capital letter N␣
↪with tilde, U+00D1 ISOlat1 -->
225                 <entity name="Ograve" cdata="&amp;#210;"/> <!--latin capital letter O␣
↪with grave, U+00D2 ISOlat1 -->
226                 <entity name="Oacute" cdata="&amp;#211;"/> <!--latin capital letter O␣
↪with acute, U+00D3 ISOlat1 -->
227                 <entity name="Ocirc" cdata="&amp;#212;"/> <!--latin capital letter O␣
↪with circumflex, U+00D4 ISOlat1 -->
228                 <entity name="Otilde" cdata="&amp;#213;"/> <!--latin capital letter O␣
↪with tilde, U+00D5 ISOlat1 -->
229                 <entity name="Ouml" cdata="&amp;#214;"/> <!--latin capital letter O␣
↪with diaeresis, U+00D6 ISOlat1 -->
230                 <entity name="times" cdata="&amp;#215;"/> <!--multiplication sign,␣
↪U+00D7 ISOnum -->
231                 <entity name="Oslash" cdata="&amp;#216;"/> <!--latin capital letter O␣
↪with stroke = latin capital letter O slash, U+00D8 ISOlat1 -->
232                 <entity name="Ugrave" cdata="&amp;#217;"/> <!--latin capital letter U␣
↪with grave, U+00D9 ISOlat1 -->
233                 <entity name="Uacute" cdata="&amp;#218;"/> <!--latin capital letter U␣
↪with acute, U+00DA ISOlat1 -->
234                 <entity name="Ucirc" cdata="&amp;#219;"/> <!--latin capital letter U␣
↪with circumflex, U+00DB ISOlat1 -->
235                 <entity name="Uuml" cdata="&amp;#220;"/> <!--latin capital letter U␣
↪with diaeresis, U+00DC ISOlat1 -->
```

(suite sur la page suivante)

(suite de la page précédente)

```
236                 <entity name="Yacute" cdata="&amp;#221;"/> <!--latin capital letter Y␣
↪with acute, U+00DD ISOlat1 -->
237                 <entity name="THORN" cdata="&amp;#222;"/> <!--latin capital letter␣
↪THORN, U+00DE ISOlat1 -->
238                 <entity name="szlig" cdata="&amp;#223;"/> <!--latin small letter␣
↪sharp s = ess-zed, U+00DF ISOlat1 -->
239                 <entity name="agrave" cdata="&amp;#224;"/> <!--latin small letter a␣
↪with grave = latin small letter a grave, U+00E0 ISOlat1 -->
240                 <entity name="aacute" cdata="&amp;#225;"/> <!--latin small letter a␣
↪with acute, U+00E1 ISOlat1 -->
241                 <entity name="acirc" cdata="&amp;#226;"/> <!--latin small letter a␣
↪with circumflex, U+00E2 ISOlat1 -->
242                 <entity name="atilde" cdata="&amp;#227;"/> <!--latin small letter a␣
↪with tilde, U+00E3 ISOlat1 -->
243                 <entity name="auml" cdata="&amp;#228;"/> <!--latin small letter a␣
↪with diaeresis, U+00E4 ISOlat1 -->
244                 <entity name="aring" cdata="&amp;#229;"/> <!--latin small letter a␣
↪with ring above = latin small letter a ring, U+00E5 ISOlat1 -->
245                 <entity name="aelig" cdata="&amp;#230;"/> <!--latin small letter ae =␣
↪latin small ligature ae, U+00E6 ISOlat1 -->
246                 <entity name="ccedil" cdata="&amp;#231;"/> <!--latin small letter c␣
↪with cedilla, U+00E7 ISOlat1 -->
247                 <entity name="egrave" cdata="&amp;#232;"/> <!--latin small letter e␣
↪with grave, U+00E8 ISOlat1 -->
248                 <entity name="eacute" cdata="&amp;#233;"/> <!--latin small letter e␣
↪with acute, U+00E9 ISOlat1 -->
249                 <entity name="ecirc" cdata="&amp;#234;"/> <!--latin small letter e␣
↪with circumflex, U+00EA ISOlat1 -->
250                 <entity name="euml" cdata="&amp;#235;"/> <!--latin small letter e␣
↪with diaeresis, U+00EB ISOlat1 -->
251                 <entity name="igrave" cdata="&amp;#236;"/> <!--latin small letter i␣
↪with grave, U+00EC ISOlat1 -->
252                 <entity name="iacute" cdata="&amp;#237;"/> <!--latin small letter i␣
↪with acute, U+00ED ISOlat1 -->
253                 <entity name="icirc" cdata="&amp;#238;"/> <!--latin small letter i␣
↪with circumflex, U+00EE ISOlat1 -->
254                 <entity name="iuml" cdata="&amp;#239;"/> <!--latin small letter i␣
↪with diaeresis, U+00EF ISOlat1 -->
255                 <entity name="eth" cdata="&amp;#240;"/> <!--latin small letter eth,␣
↪U+00F0 ISOlat1 -->
256                 <entity name="ntilde" cdata="&amp;#241;"/> <!--latin small letter n␣
↪with tilde, U+00F1 ISOlat1 -->
257                 <entity name="ograve" cdata="&amp;#242;"/> <!--latin small letter o␣
↪with grave, U+00F2 ISOlat1 -->
258                 <entity name="oacute" cdata="&amp;#243;"/> <!--latin small letter o␣
↪with acute, U+00F3 ISOlat1 -->
259                 <entity name="ocirc " cdata="&amp;#244;"/> <!--latin small letter o␣
↪with circumflex, U+00F4 ISOlat1 -->
260                 <entity name="otilde" cdata="&amp;#245;"/> <!--latin small letter o␣
↪with tilde, U+00F5 ISOlat1 -->
261                 <entity name="ouml" cdata="&amp;#246;"/> <!--latin small letter o␣
↪with diaeresis, U+00F6 ISOlat1 -->
262                 <entity name="divide" cdata="&amp;#247;"/> <!--division sign, U+00F7␣
↪ISOnum -->
263                 <entity name="oslash" cdata="&amp;#248;"/> <!--latin small letter o␣
↪with stroke, = latin small letter o slash, U+00F8 ISOlat1 -->
264                 <entity name="ugrave" cdata="&amp;#249;"/> <!--latin small letter u␣
↪with grave, U+00F9 ISOlat1 -->
```

(suite sur la page suivante)

(suite de la page précédente)

```
265                 <entity name="uacute" cdata="&amp;#250;"/> <!--latin small letter u␣
↪with acute, U+00FA ISOlat1 -->
266                 <entity name="ucirc" cdata="&amp;#251;"/> <!--latin small letter u␣
↪with circumflex, U+00FB ISOlat1 -->
267                 <entity name="uuml" cdata="&amp;#252;"/> <!--latin small letter u␣
↪with diaeresis, U+00FC ISOlat1 -->
268                 <entity name="yacute" cdata="&amp;#253;"/> <!--latin small letter y␣
↪with acute, U+00FD ISOlat1 -->
269                 <entity name="thorn" cdata="&amp;#254;"/> <!--latin small letter␣
↪thorn, U+00FE ISOlat1 -->
270                 <entity name="yuml" cdata="&amp;#255;"/> <!--latin small letter y␣
↪with diaeresis, U+00FF ISOlat1 -->
271
272                 <entity name="fnof" cdata="&amp;#402;"/> <!--latin small f with hook␣
↪= function = florin, U+0192 ISOtech -->
273
274                 <!-- Greek -->
275                 <entity name="Alpha" cdata="&amp;#913;"/> <!--greek capital letter␣
↪alpha, U+0391 -->
276                 <entity name="Beta" cdata="&amp;#914;"/> <!--greek capital letter␣
↪beta, U+0392 -->
277                 <entity name="Gamma" cdata="&amp;#915;"/> <!--greek capital letter␣
↪gamma, U+0393 ISOgrk3 -->
278                 <entity name="Delta" cdata="&amp;#916;"/> <!--greek capital letter␣
↪delta, U+0394 ISOgrk3 -->
279                 <entity name="Epsilon" cdata="&amp;#917;"/> <!--greek capital letter␣
↪epsilon, U+0395 -->
280                 <entity name="Zeta" cdata="&amp;#918;"/> <!--greek capital letter␣
↪zeta, U+0396 -->
281                 <entity name="Eta" cdata="&amp;#919;"/> <!--greek capital letter eta,␣
↪U+0397 -->
282                 <entity name="Theta" cdata="&amp;#920;"/> <!--greek capital letter␣
↪theta, U+0398 ISOgrk3 -->
283                 <entity name="Iota" cdata="&amp;#921;"/> <!--greek capital letter␣
↪iota, U+0399 -->
284                 <entity name="Kappa" cdata="&amp;#922;"/> <!--greek capital letter␣
↪kappa, U+039A -->
285                 <entity name="Lambda" cdata="&amp;#923;"/> <!--greek capital letter␣
↪lambda, U+039B ISOgrk3 -->
286                 <entity name="Mu" cdata="&amp;#924;"/> <!--greek capital letter mu,␣
↪U+039C -->
287                 <entity name="Nu" cdata="&amp;#925;"/> <!--greek capital letter nu,␣
↪U+039D -->
288                 <entity name="Xi" cdata="&amp;#926;"/> <!--greek capital letter xi,␣
↪U+039E ISOgrk3 -->
289                 <entity name="Omicron" cdata="&amp;#927;"/> <!--greek capital letter␣
↪omicron, U+039F -->
290                 <entity name="Pi" cdata="&amp;#928;"/> <!--greek capital letter pi,␣
↪U+03A0 ISOgrk3 -->
291                 <entity name="Rho" cdata="&amp;#929;"/> <!--greek capital letter rho,␣
↪U+03A1 -->
292                 <!-- there is no Sigmaf, and no U+03A2 character either -->
293                 <entity name="Sigma" cdata="&amp;#931;"/> <!--greek capital letter␣
↪sigma, U+03A3 ISOgrk3 -->
294                 <entity name="Tau" cdata="&amp;#932;"/> <!--greek capital letter tau,␣
↪U+03A4 -->
295                 <entity name="Upsilon" cdata="&amp;#933;"/> <!--greek capital letter␣
↪upsilon,U+03A5 ISOgrk3 -->
```

(suite sur la page suivante)

(suite de la page précédente)

```
296                 <entity name="Phi" cdata="&amp;#934;"/> <!--greek capital letter phi,
↪U+03A6 ISOgrk3 -->
297                 <entity name="Chi" cdata="&amp;#935;"/> <!--greek capital letter chi,␣
↪U+03A7 -->
298                 <entity name="Psi" cdata="&amp;#936;"/> <!--greek capital letter psi,
↪U+03A8 ISOgrk3 -->
299                 <entity name="Omega" cdata="&amp;#937;"/> <!--greek capital letter␣
↪omega,U+03A9 ISOgrk3 -->
300
301                 <entity name="alpha" cdata="&amp;#945;"/> <!--greek small letter␣
↪alpha,U+03B1 ISOgrk3 -->
302                 <entity name="beta" cdata="&amp;#946;"/> <!--greek small letter beta,␣
↪U+03B2 ISOgrk3 -->
303                 <entity name="gamma" cdata="&amp;#947;"/> <!--greek small letter␣
↪gamma,U+03B3 ISOgrk3 -->
304                 <entity name="delta" cdata="&amp;#948;"/> <!--greek small letter␣
↪delta,U+03B4 ISOgrk3 -->
305                 <entity name="epsilon" cdata="&amp;#949;"/> <!--greek small letter␣
↪epsilon,U+03B5 ISOgrk3 -->
306                 <entity name="zeta" cdata="&amp;#950;"/> <!--greek small letter zeta,␣
↪U+03B6 ISOgrk3 -->
307                 <entity name="eta" cdata="&amp;#951;"/> <!--greek small letter eta,␣
↪U+03B7 ISOgrk3 -->
308                 <entity name="theta" cdata="&amp;#952;"/> <!--greek small letter␣
↪theta, U+03B8 ISOgrk3 -->
309                 <entity name="iota" cdata="&amp;#953;"/> <!--greek small letter iota,␣
↪U+03B9 ISOgrk3 -->
310                 <entity name="kappa" cdata="&amp;#954;"/> <!--greek small letter␣
↪kappa,U+03BA ISOgrk3 -->
311                 <entity name="lambda" cdata="&amp;#955;"/> <!--greek small letter␣
↪lambda, U+03BB ISOgrk3 -->
312                 <entity name="mu" cdata="&amp;#956;"/> <!--greek small letter mu,␣
↪U+03BC ISOgrk3 -->
313                 <entity name="nu" cdata="&amp;#957;"/> <!--greek small letter nu,␣
↪U+03BD ISOgrk3 -->
314                 <entity name="xi" cdata="&amp;#958;"/> <!--greek small letter xi,␣
↪U+03BE ISOgrk3 -->
315                 <entity name="omicron" cdata="&amp;#959;"/> <!--greek small letter␣
↪omicron, U+03BF NEW -->
316                 <entity name="pi" cdata="&amp;#960;"/> <!--greek small letter pi,␣
↪U+03C0 ISOgrk3 -->
317                 <entity name="rho" cdata="&amp;#961;"/> <!--greek small letter rho,␣
↪U+03C1 ISOgrk3 -->
318                 <entity name="sigmaf" cdata="&amp;#962;"/> <!--greek small letter␣
↪final sigma, U+03C2 ISOgrk3 -->
319                 <entity name="sigma" cdata="&amp;#963;"/> <!--greek small letter␣
↪sigma, U+03C3 ISOgrk3 -->
320                 <entity name="tau" cdata="&amp;#964;"/> <!--greek small letter tau,␣
↪U+03C4 ISOgrk3 -->
321                 <entity name="upsilon" cdata="&amp;#965;"/> <!--greek small letter␣
↪upsilon, U+03C5 ISOgrk3 -->
322                 <entity name="phi" cdata="&amp;#966;"/> <!--greek small letter phi,␣
↪U+03C6 ISOgrk3 -->
323                 <entity name="chi" cdata="&amp;#967;"/> <!--greek small letter chi,␣
↪U+03C7 ISOgrk3 -->
324                 <entity name="psi" cdata="&amp;#968;"/> <!--greek small letter psi,␣
↪U+03C8 ISOgrk3 -->
```

(suite sur la page suivante)

(suite de la page précédente)

```
<entity name="omega" cdata="&amp;#969;"/> <!--greek small letter omega, U+03C9 ISOgrk3 -->
<entity name="thetasym" cdata="&amp;#977;"/> <!--greek small letter theta symbol, U+03D1 NEW -->
<entity name="upsih" cdata="&amp;#978;"/> <!--greek upsilon with hook symbol, U+03D2 NEW -->
<entity name="piv" cdata="&amp;#982;"/> <!--greek pi symbol, U+03D6 ISOgrk3 -->

<!-- General Punctuation -->
<entity name="bull" cdata="&amp;#8226;"/> <!--bullet = black small circle, U+2022 ISOpub  -->
<!-- bullet is NOT the same as bullet operator, U+2219 -->
<entity name="hellip" cdata="&amp;#8230;"/> <!--horizontal ellipsis = three dot leader, U+2026 ISOpub  -->
<entity name="prime" cdata="&amp;#8242;"/> <!--prime = minutes = feet, U+2032 ISOtech -->
<entity name="Prime" cdata="&amp;#8243;"/> <!--double prime = seconds = inches, U+2033 ISOtech -->
<entity name="oline" cdata="&amp;#8254;"/> <!--overline = spacing overscore, U+203E NEW -->
<entity name="frasl" cdata="&amp;#8260;"/> <!--fraction slash, U+2044 NEW -->

<!-- Letterlike Symbols -->
<entity name="weierp" cdata="&amp;#8472;"/> <!--script capital P = power set = Weierstrass p, U+2118 ISOamso -->
<entity name="image" cdata="&amp;#8465;"/> <!--blackletter capital I = imaginary part, U+2111 ISOamso -->
<entity name="real" cdata="&amp;#8476;"/> <!--blackletter capital R = real part symbol, U+211C ISOamso -->
<entity name="trade" cdata="&amp;#8482;"/> <!--trade mark sign, U+2122 ISOnum -->
<entity name="alefsym" cdata="&amp;#8501;"/> <!--alef symbol = first transfinite cardinal, U+2135 NEW -->
<!-- alef symbol is NOT the same as hebrew letter alef,
     U+05D0 although the same glyph could be used to depict both characters -->

<!-- Arrows -->
<entity name="larr" cdata="&amp;#8592;"/> <!--leftwards arrow, U+2190 ISOnum -->
<entity name="uarr" cdata="&amp;#8593;"/> <!--upwards arrow, U+2191 ISOnum-->
<entity name="rarr" cdata="&amp;#8594;"/> <!--rightwards arrow, U+2192 ISOnum -->
<entity name="darr" cdata="&amp;#8595;"/> <!--downwards arrow, U+2193 ISOnum -->
<entity name="harr" cdata="&amp;#8596;"/> <!--left right arrow, U+2194 ISOamsa -->
<entity name="crarr" cdata="&amp;#8629;"/> <!--downwards arrow with corner leftwards
                                                = carriage return, U+21B5 NEW -->
<entity name="lArr" cdata="&amp;#8656;"/> <!--leftwards double arrow, U+21D0 ISOtech -->

<!-- ISO 10646 does not say that lArr is the same as the 'is implied by' arrow
```

(suite sur la page suivante)

(suite de la page précédente)

```
                    but also does not have any other character for that function. So ? lArr can
                    be used for 'is implied by' as ISOtech suggests -->

                <entity name="uArr" cdata="&amp;#8657;"/> <!--upwards double arrow, U+21D1 ISOamsa -->
                <entity name="rArr" cdata="&amp;#8658;"/> <!--rightwards double arrow, U+21D2 ISOtech -->

                <!-- ISO 10646 does not say this is the 'implies' character but does not have
                     another character with this function so ?
                     rArr can be used for 'implies' as ISOtech suggests -->

                <entity name="dArr" cdata="&amp;#8659;"/> <!--downwards double arrow, U+21D3 ISOamsa -->
                <entity name="hArr" cdata="&amp;#8660;"/> <!--left right double arrow, U+21D4 ISOamsa -->

                <!-- Mathematical Operators -->
                <entity name="forall" cdata="&amp;#8704;"/> <!--for all, U+2200 ISOtech -->
                <entity name="part" cdata="&amp;#8706;"/> <!--partial differential, U+2202 ISOtech  -->
                <entity name="exist" cdata="&amp;#8707;"/> <!--there exists, U+2203 ISOtech -->
                <entity name="empty" cdata="&amp;#8709;"/> <!--empty set = null set = diameter,U+2205 ISOamso -->
                <entity name="nabla" cdata="&amp;#8711;"/> <!--nabla = backward difference, U+2207 ISOtech -->
                <entity name="isin" cdata="&amp;#8712;"/> <!--element of, U+2208 ISOtech -->
                <entity name="notin" cdata="&amp;#8713;"/> <!--not an element of, U+2209 ISOtech -->
                <entity name="ni" cdata="&amp;#8715;"/> <!--contains as member, U+220B ISOtech -->

                <!-- should there be a more memorable name than 'ni'? -->
                <entity name="prod" cdata="&amp;#8719;"/> <!--n-ary product = product sign, U+220F ISOamsb -->

                <!-- prod is NOT the same character as U+03A0 'greek capital letter pi ' though
                     the same glyph might be used for both -->

                <entity name="sum" cdata="&amp;#8721;"/> <!--n-ary sumation, U+2211 ISOamsb -->

                <!-- sum is NOT the same character as U+03A3 'greek capital letter sigma'
                     though the same glyph might be used for both -->

                <entity name="minus" cdata="&amp;#8722;"/> <!--minus sign, U+2212 ISOtech -->
                <entity name="lowast" cdata="&amp;#8727;"/> <!--asterisk operator, U+2217 ISOtech -->
                <entity name="radic" cdata="&amp;#8730;"/> <!--square root = radical sign, U+221A ISOtech -->
```

(suite sur la page suivante)

(suite de la page précédente)

```
396                 <entity name="prop" cdata="&amp;#8733;"/> <!--proportional to, U+221D␣
↪ISOtech -->
397                 <entity name="infin" cdata="&amp;#8734;"/> <!--infinity, U+221E␣
↪ISOtech -->
398                 <entity name="ang" cdata="&amp;#8736;"/> <!--angle, U+2220 ISOamso -->
399                 <entity name="and" cdata="&amp;#8743;"/> <!--logical and = wedge,␣
↪U+2227 ISOtech -->
400                 <entity name="or" cdata="&amp;#8744;"/> <!--logical or = vee, U+2228␣
↪ISOtech -->
401                 <entity name="cap" cdata="&amp;#8745;"/> <!--intersection = cap,␣
↪U+2229 ISOtech -->
402                 <entity name="cup" cdata="&amp;#8746;"/> <!--union = cup, U+222A␣
↪ISOtech -->
403                 <entity name="int" cdata="&amp;#8747;"/> <!--integral, U+222B ISOtech␣
↪-->
404                 <entity name="there4" cdata="&amp;#8756;"/> <!--therefore, U+2234␣
↪ISOtech -->
405                 <entity name="sim" cdata="&amp;#8764;"/> <!--tilde operator = varies␣
↪with = similar to, U+223C ISOtech -->
406
407                 <!-- tilde operator is NOT the same character as the tilde, U+007E,
408                      although the same glyph might be used to represent both  -->
409
410                 <entity name="cong" cdata="&amp;#8773;"/> <!--approximately equal to,␣
↪U+2245 ISOtech -->
411                 <entity name="asymp" cdata="&amp;#8776;"/> <!--almost equal to =␣
↪asymptotic to, U+2248 ISOamsr -->
412                 <entity name="ne" cdata="&amp;#8800;"/> <!--not equal to, U+2260␣
↪ISOtech -->
413                 <entity name="equiv" cdata="&amp;#8801;"/> <!--identical to, U+2261␣
↪ISOtech -->
414                 <entity name="le" cdata="&amp;#8804;"/> <!--less-than or equal to,␣
↪U+2264 ISOtech -->
415                 <entity name="ge" cdata="&amp;#8805;"/> <!--greater-than or equal to,␣
↪U+2265 ISOtech -->
416                 <entity name="sub" cdata="&amp;#8834;"/> <!--subset of, U+2282␣
↪ISOtech -->
417                 <entity name="sup" cdata="&amp;#8835;"/> <!--superset of, U+2283␣
↪ISOtech -->
418
419                 <!-- note that nsup, 'not a superset of, U+2283' is not covered by␣
↪the Symbol
420                      font encoding and is not included. Should it be, for symmetry?
421                      It is in ISOamsn  -->
422
423                 <entity name="nsub" cdata="&amp;#8836;"/> <!--not a subset of, U+2284␣
↪ISOamsn -->
424                 <entity name="sube" cdata="&amp;#8838;"/> <!--subset of or equal to,␣
↪U+2286 ISOtech -->
425                 <entity name="supe" cdata="&amp;#8839;"/> <!--superset of or equal to,
↪ U+2287 ISOtech -->
426                 <entity name="oplus" cdata="&amp;#8853;"/> <!--circled plus = direct␣
↪sum, U+2295 ISOamsb -->
427                 <entity name="otimes" cdata="&amp;#8855;"/> <!--circled times =␣
↪vector product, U+2297 ISOamsb -->
428                 <entity name="perp" cdata="&amp;#8869;"/> <!--up tack = orthogonal to␣
↪= perpendicular, U+22A5 ISOtech -->
```

(suite sur la page suivante)

(suite de la page précédente)

```
429                 <entity name="sdot" cdata="&amp;#8901;"/> <!--dot operator, U+22C5␣
↪ISOamsb -->
430                 <!-- dot operator is NOT the same character as U+00B7 middle dot -->
431
432                 <!-- Miscellaneous Technical -->
433                 <entity name="lceil" cdata="&amp;#8968;"/> <!--left ceiling = apl␣
↪upstile, U+2308 ISOamsc  -->
434                 <entity name="rceil" cdata="&amp;#8969;"/> <!--right ceiling, U+2309␣
↪ISOamsc  -->
435                 <entity name="lfloor" cdata="&amp;#8970;"/> <!--left floor = apl␣
↪downstile, U+230A ISOamsc  -->
436                 <entity name="rfloor" cdata="&amp;#8971;"/> <!--right floor, U+230B␣
↪ISOamsc  -->
437                 <entity name="lang" cdata="&amp;#9001;"/> <!--left-pointing angle␣
↪bracket = bra, U+2329 ISOtech -->
438                 <!-- lang is NOT the same character as U+003C 'less than'
439                      or U+2039 'single left-pointing angle quotation mark' -->
440                 <entity name="rang" cdata="&amp;#9002;"/> <!--right-pointing angle␣
↪bracket = ket, U+232A ISOtech -->
441                 <!-- rang is NOT the same character as U+003E 'greater than' or␣
↪U+203A 'single right-pointing angle quotation mark' -->
442
443                 <!-- Geometric Shapes -->
444                 <entity name="loz" cdata="&amp;#9674;"/> <!--lozenge, U+25CA ISOpub --
↪>
445
446                 <!-- Miscellaneous Symbols -->
447                 <entity name="spades" cdata="&amp;#9824;"/> <!--black spade suit,␣
↪U+2660 ISOpub -->
448                 <!-- black here seems to mean filled as opposed to hollow -->
449                 <entity name="clubs" cdata="&amp;#9827;"/> <!--black club suit =␣
↪shamrock, U+2663 ISOpub -->
450                 <entity name="hearts" cdata="&amp;#9829;"/> <!--black heart suit =␣
↪valentine, U+2665 ISOpub -->
451                 <entity name="diams" cdata="&amp;#9830;"/> <!--black diamond suit,␣
↪U+2666 ISOpub -->
452
453                 <entity name="quot" cdata="&amp;#34;"  /> <!--quotation mark = APL␣
↪quote, U+0022 ISOnum -->
454                 <!-- Latin Extended-A -->
455                 <entity name="OElig" cdata="&amp;#338;" /> <!--latin capital ligature␣
↪OE, U+0152 ISOlat2 -->
456                 <entity name="oelig" cdata="&amp;#339;" /> <!--latin small ligature␣
↪oe, U+0153 ISOlat2 -->
457                 <!-- ligature is a misnomer, this is a separate character in some␣
↪languages -->
458                 <entity name="Scaron" cdata="&amp;#352;" /> <!--latin capital letter␣
↪S with caron, U+0160 ISOlat2 -->
459                 <entity name="scaron" cdata="&amp;#353;" /> <!--latin small letter s␣
↪with caron, U+0161 ISOlat2 -->
460                 <entity name="Yuml" cdata="&amp;#376;" /> <!--latin capital letter Y␣
↪with diaeresis, U+0178 ISOlat2 -->
461
462                 <!-- Spacing Modifier Letters -->
463                 <entity name="circ" cdata="&amp;#710;" /> <!--modifier letter␣
↪circumflex accent, U+02C6 ISOpub -->
464                 <entity name="tilde" cdata="&amp;#732;" /> <!--small tilde, U+02DC␣
↪ISOdia -->
```

(suite sur la page suivante)

(suite de la page précédente)

```
                <!-- General Punctuation -->
                <entity name="ensp" cdata="&amp;#8194;"/> <!--en space, U+2002 ISOpub␣
↪-->
                <entity name="emsp" cdata="&amp;#8195;"/> <!--em space, U+2003 ISOpub␣
↪-->
                <entity name="thinsp" cdata="&amp;#8201;"/> <!--thin space, U+2009␣
↪ISOpub -->
                <entity name="zwnj" cdata="&amp;#8204;"/> <!--zero width non-joiner,␣
↪U+200C NEW RFC 2070 -->
                <entity name="zwj" cdata="&amp;#8205;"/> <!--zero width joiner,␣
↪U+200D NEW RFC 2070 -->
                <entity name="lrm" cdata="&amp;#8206;"/> <!--left-to-right mark,␣
↪U+200E NEW RFC 2070 -->
                <entity name="rlm" cdata="&amp;#8207;"/> <!--right-to-left mark,␣
↪U+200F NEW RFC 2070 -->
                <entity name="ndash" cdata="&amp;#8211;"/> <!--en dash, U+2013 ISOpub␣
↪-->
                <entity name="mdash" cdata="&amp;#8212;"/> <!--em dash, U+2014 ISOpub␣
↪-->
                <entity name="lsquo" cdata="&amp;#8216;"/> <!--left single quotation␣
↪mark, U+2018 ISOnum -->
                <entity name="rsquo" cdata="&amp;#8217;"/> <!--right single quotation␣
↪mark, U+2019 ISOnum -->
                <entity name="sbquo" cdata="&amp;#8218;"/> <!--single low-9 quotation␣
↪mark, U+201A NEW -->
                <entity name="ldquo" cdata="&amp;#8220;"/> <!--left double quotation␣
↪mark, U+201C ISOnum -->
                <entity name="rdquo" cdata="&amp;#8221;"/> <!--right double quotation␣
↪mark, U+201D ISOnum -->
                <entity name="bdquo" cdata="&amp;#8222;"/> <!--double low-9 quotation␣
↪mark, U+201E NEW -->
                <entity name="dagger" cdata="&amp;#8224;"/> <!--dagger, U+2020 ISOpub␣
↪-->
                <entity name="Dagger" cdata="&amp;#8225;"/> <!--double dagger, U+2021␣
↪ISOpub -->
                <entity name="permil" cdata="&amp;#8240;"/> <!--per mille sign,␣
↪U+2030 ISOtech -->
                <entity name="lsaquo" cdata="&amp;#8249;"/> <!--single left-pointing␣
↪angle quotation mark, U+2039 ISO proposed -->
                <!-- lsaquo is proposed but not yet ISO standardized -->
                <entity name="rsaquo" cdata="&amp;#8250;"/> <!--single right-pointing␣
↪angle quotation mark, U+203A ISO proposed -->
                <!-- rsaquo is proposed but not yet ISO standardized -->
                <entity name="euro" cdata="&amp;#8364;" /> <!--euro sign, U+20AC NEW -
↪->
        </html-entities>

</anti-samy-rules>
```

8.2.1.8 Fichier `/vitam/conf/<composant>/vitam.conf`

```
secret : {{ plateforme_secret }}
filterActivation : {{ vitam_struct.secret_platform }}
{% if vitam_struct.vitam_component == vitam.processing.vitam_component %}
```

(suite sur la page suivante)

(suite de la page précédente)

```
distributeurBatchSize: 800
workerBulkSize: 16
{% endif %}
{% if vitam_struct.vitam_component == vitam.metadata.vitam_component %}
storeGraphElementsPerFile: 10000
storeGraphOverlapDelay: 300
expireCacheEntriesDelay: 300
deleteIncompleteReconstructedUnitDelay: 2592000
migrationBulkSize: 10000
{% endif %}
distributionThreshold : 100000
eliminationAnalysisThreshold : 100000
eliminationActionThreshold : 10000
intervalDelayCheckIdle : 5000
maxDelayUnusedConnection : 5000
delayValidationAfterInactivity : 2500
tenants:  [ "{{ vitam_tenant_ids | join('", "') }}" ]
adminTenant : {{ vitam_tenant_admin }}
forceChunkModeInputStream : {{ vitam_defaults.vitam_force_chunk_mode }}

{% if vitam_struct.vitam_component == vitam.worker.vitam_component %}
reclassificationMaxBulkThreshold: 1000
reclassificationMaxUnitsThreshold: 10000
reclassificationMaxGuildListSizeInLogbookOperation: 1000
{% endif %}

keywordMaxLength: 32766
textMaxLength: 32766

classificationLevel :
  allowList :  [{% for classification in classificationList %}{{ classification }}{%
↪if not loop.last %},{% endif %}{% endfor %}]
  authorizeNotDefined: {{ classificationLevelOptional }}

indexInheritedRulesWithAPIV2OutputByTenant: [ "{{  vitam.worker.api_output_index_
↪tenants | join('", "') }}" ]
indexInheritedRulesWithRulesIdByTenant: [ "{{  vitam.worker.rules_index_tenants |
↪join('", "') }}" ]

environmentName: {{ vitam_prefix_offer|default(vitam_site_name) }}

acceptableRequestTime: {{ vitam_struct.vitam_component.
↪acceptableRequestTime|default(10) }}

# Ontology cache settings (max entries in cache & retention timeout in seconds)
ontologyCacheMaxEntries: {{ vitam.ontologyCacheMaxEntries }}

# Elasticsearch scroll timeout settings
elasticSearchScrollTimeoutInMilliseconds: {{ vitam.
↪elasticSearchScrollTimeoutInMilliseconds }}
```

Ce fichier permet de définir le secret de plate-forme.

8.2.1.9 Fichier /vitam/conf/<composant>/vitam.metrics.conf

```
#  Fichier de configuration des métriques
#
#  Les différents clés disponibles pour ce fichier de configuration sont les suivantes :
#
#  metricsJersey: true / false        Active ou non les métriques Jersey
#  metricsJVM: true / false        Active ou non les métriques JVM
#
#  metricReporter: ELASTICSEARCH | LOGBACK | NONE                          défini le type de reporter
#  metricReporterInterval: int > 0                                          défini l'interval entre chaque reporting
#  metricReporterIntervalUnit: TimeUnit (ex: SECONDS, MINUTES...)        défini le type d'interval
#
#  Si le reporter est de type LOGBACK, la clé suivante est configurable:
#  metricLogLevel: DEBUG | INFO | WARN | ERROR ...                          défini le niveau de log Logback
#
#  Si le reporter est de type ELASTICSEARCH, la clé suivante est obligatoire :
#
#  (un tableau avec les différentes adresses des bases ElasticSearch)
#  metricReporterHosts:
#          - 127.0.0.1:9201
#          - 0.0.0.0:80
#          - 8.8.8.8:22

{% if (groups['hosts_elasticsearch_log'] | length) > 0  %}
metricsJersey: true
metricsJVM: true

metricReporter: ELASTICSEARCH
metricReporterHosts:
{% for host in groups['hosts_elasticsearch_log'] %}
   - "{{ hostvars[host]['ip_admin'] }}:{{ elasticsearch.log.port_http }}"
{% endfor %}
metricLogLevel: {{ vitam_struct.metricslevel|default('DEBUG') }}
metricReporterInterval: {{ vitam_struct.metricsinterval|default('3') }}
metricReporterIntervalUnit: {{ vitam_struct.metricsunit|default('MINUTES') }}
{% endif %}
```

8.2.1.10 Fichier /vitam/conf/<composant>/java.security

```
# Use Bouncy Castle Provider when it is available
security.provider.9=org.bouncycastle.jce.provider.BouncyCastleProvider

# Override the default list of Centos 7 that disable Elliptic Curved Based Algorithms
jdk.tls.disabledAlgorithms="SSLv3, RC4, MD5withRSA, DH keySize < 768,RSA keySize < 2048"
```

8.2.2 Access

8.2.2.1 access external

8.2.2.1.1 Présentation

Access-external est le composant d'interface entre *VITAM* et un *SIA* client, permettant de réaliser des recherches sur les objets archivés et les journaux. Il permet également quelques fonctions d'administration, en particulier les chargements des référentiels.

Rôle :

- Exposer les API publiques du système
- Sécuriser l'accès aux API de VITAM

8.2.2.1.2 Configuration / fichiers utiles

Les fichiers de configuration sont gérés par les procédures d'installation ou de mise à niveau de l'environnement *VITAM*. Se référer au *DIN*.

Les fichiers de configuration sont définis sous `/vitam/conf/access-external`.

8.2.2.1.2.1 Fichier `access-external.conf`

```
authentication: false
jettyConfig: jetty-config.xml
tenantFilter : true
```

8.2.2.1.2.2 Fichier `access-internal-client.conf`

```
serverHost: {{ vitam.accessinternal.host }}
serverPort: {{ vitam.accessinternal.port_service }}
```

8.2.2.1.2.3 Fichier `functional-administration-client.conf`

```
serverHost: {{ vitam.functional_administration.host }}
serverPort: {{ vitam.functional_administration.port_service }}
```

8.2.2.1.2.4 Fichier `ingest-internal-client.conf`

```
serverHost: {{ vitam.ingestinternal.host }}
serverPort: {{ vitam.ingestinternal.port_service }}
```

8.2.2.1.2.5 Fichier `internal-security-client.conf`

```
serverHost: {{ vitam.security_internal.host }}
serverPort: {{ vitam.security_internal.port_service }}
secure: false
```

8.2.2.1.3 Opérations

- Démarrage du service

En tant qu'utilisateur root : `systemctl start vitam-access-external`

- Arrêt du service

En tant qu'utilisateur root : `systemctl stop vitam-access-external`

- Sauvegarde du service

Ce service ne nécessite pas de sauvegarde particulière.

- Supervision du service

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port>/access-external/v1/status`

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port admin>/admin/v1/status`

- Exports

N/A

- gestion de la capacité

N/A

- actions récurrentes

- cas des batches

N/A

8.2.2.2 access-internal

8.2.2.2.1 Présentation du composant

Access-internal est le composant *VITAM*, permettant de réaliser des recherches et consultations sur les objets archivés et les journaux. Il permet également de modifier les informations d'un *ArchiveUnit*.

Rôle :

- Permettre l'accès aux données du système VITAM

Fonction :

- Exposition des fonctions de recherche d'archives offertes par metadata ;
- Exposition des fonctions de parcours de journaux offertes par logbook.

8.2.2.2.2 Configuration / fichiers utiles

Les fichiers de configuration sont gérés par les procédures d'installation ou de mise à niveau de l'environnement *VITAM*. Se référer au *DIN*.

Les fichiers de configuration sont définis sous `/vitam/conf/access`.

8.2.2.2.2.1 Fichier `access-internal.conf`

Ce fichier permet de définir l'URL d'accès au metadata server.

```
urlMetaData: {{vitam.metadata | client_url}}
urlWorkspace: {{vitam.workspace | client_url}}
urlProcessing: {{vitam.processing | client_url}}
jettyConfig: jetty-config.xml
```

8.2.2.2.2.2 Fichier `storage-client.conf`

Ce fichier permet de définir l'accès au storage-engine.

```
serverHost: {{ vitam.storageengine.host }}
serverPort: {{ vitam.storageengine.port_service }}
```

8.2.2.2.2.3 Fichier `metadata-client.conf`

```
serverHost: {{ vitam.metadata.host }}
serverPort: {{ vitam.metadata.port_service }}
```

8.2.2.2.2.4 Fichier `functional-administration-client.conf`

```
serverHost: {{ vitam.functional_administration.host }}
serverPort: {{ vitam.functional_administration.port_service }}
```

8.2.2.2.3 Opérations

- Démarrage du service

En tant qu'utilisateur root : `systemctl start vitam-access-internal`

- Arrêt du service

En tant qu'utilisateur root : `systemctl stop vitam-access-internal`

- Sauvegarde du service

Ce service ne nécessite pas de sauvegarde particulière.

- Supervision du service

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port>/access/v1/status`

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port admin>/admin/v1/status`

- Exports

N/A

- gestion de la capacité

N/A

- actions récurrentes

- cas des batches

N/A

8.2.3 Batch-Report

8.2.3.1 Présentation

Le composant batch-report permet de stocker des données de traitements de masse (en particulier, élimination) pour les aggréger sous forme de rapports.

Ce module utilise une base de données MongoDB « report », dans laquelle sont stockées, sous différentes collections (entre autres, EliminationActionObjectGroup et EliminationActionUnit), les données.

Ce module est appelé par le composant « worker » pour collecter les données durant les *workflows* d'élimination, entre autres.

Ce module est ensuite appelé par le composant « worker » pour restituer les données aggrégées sous forme de rapports.

8.2.3.2 Configuration

Ce document spécifie la configuration (fichiers de config) pour lancer le services de batch-report.

Tous ces fichiers de configuration sont placés dans le répertoire `/vitam/conf/batch-report`.

8.2.3.2.1 Fichier `batch-report.conf`

```
# Configuration MongoDB
mongoDbNodes:
{% for server in groups['hosts_mongos_data'] %}
- dbHost: {{ hostvars[server]['ip_service'] }}
  dbPort: {{ mongodb.mongos_port }}
{% endfor %}
dbName: report
dbAuthentication: {{ mongodb.mongo_authentication }}
dbUserName: {{ mongodb['mongo-data'].report.user }}
dbPassword: {{ mongodb['mongo-data'].report.password }}
jettyConfig: jetty-config.xml

workspaceUrl: {{ vitam.workspace | client_url }}
```

8.2.3.3 Client batch-report

Pour la création d'un client batch-report, nous avons besoin aussi du fichier de configuration `batch-report-client.conf` qui précise le serveur host et la porte du serveur où le client se connecte pour les requêtes.

8.2.3.4 Opérations

- Démarrage du service

En tant qu'utilisateur root : `systemctl start vitam-batch-report`

- Arrêt du service

En tant qu'utilisateur root : `systemctl stop vitam-batch-report`

- Sauvegarde du service

Ce service ne nécessite pas de sauvegarde particulière.

- Supervision du service

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port admin>/admin/v1/status`

- Exports

N/A

- gestion de la capacité

N/A

- actions récurrentes

- cas des batches

N/A

8.2.4 common-plugin

8.2.4.1 Présentation du composant

common-plugin est le composant permettant de réaliser des plugins sans appel à des package privé . Rôle :

- l'objet de ce common-plugin n'est pas que de fournir des interfaces à implémenter mais aussi les classes d'implémentations imposées par Vitam pour réaliser des plugins.

Fonction :

- Exposition interfaces à implémenter et les classes d'implémentations pour réaliser des plugins .

8.2.4.2 Classes utiles

L'Objectif de Plugin Common est d'inclure tous les classes utiles afin de créer un plugin à partir de ce package .

Les classes de model sont définis sous `/vitam/common/model`.

8.2.4.2.1 Classe `Item Status`

Ce classe permet de retourner le statut d'un Item.

8.2.4.2.2 Classe `VitamAutoCloseable`

Le mot clé try-with-resources garantit que chaque ressource sera fermée lorsqu'elle n'est plus utilisée. Une ressource et un objet qui implémente l'interface VitamAutoCloseable. Il est donc possible d'utiliser une instance de ces interfaces avec le mot clé try-with-resources.

Les classes de common parameter sont définis sous `/vitam/common/parameter`.

8.2.4.2.3 Classe `ParameterHelper`

Ce classe permet de faire un check sur les paramètres et avoir le tenant parameter de session vitam .

8.2.4.2.4 Classe `VitamParameter`

Cet interface permet d'aider à créer des nouveaux paramètres liés au classes .

Les classes de common exception sont définis sous `/vitam/processing/common/exception`.

8.2.4.2.5 Classe `ProcessingException`

Ce classe est le classe père de tous les Vitam Processing Exception .

Les classes de model common processing sont définis sous `/vitam/processing/common/model`.

8.2.4.2.6 Classe `IOParameter`

Ce class permet de définir les paramètres Input et Output pour une action et une step .

8.2.4.2.7 Classe `ProcessingUri`

Ce classe permet de formatter le processing URI .

8.2.4.2.8 Classe `UriPrefix`

C'est le Handler IO

Les classes des paramètres common sont définis sous `/vitam/processing/common/parameter`.

8.2.4.2.9 Classe `AbstractWorkerParameters`

C'est une implémentation abstraite de tous les paramètres de workers .

8.2.4.2.10 Classe `DefaultWorkerParameters`

Ce classe permet de définir les paramètres par défaut d'un worker.

8.2.4.2.11 Classe `WorkerParameterName`

Ce classe inclut une énumération avec tous les noms des paramètres d'un worker .

8.2.4.2.12 Classe `WorkerParameters`

Ce classe permet de définir les paramètres de worker.

8.2.4.2.13 Classe `WorkerParametersDeserializer`

Ce classe permet de définir les paramètres d'un worker deserializer.

8.2.4.2.14 Classe `WorkerParametersFactory`

Ce classe permet de définir les paramètres d'un worker Factory.

8.2.4.2.15 Classe `WorkerParametersSerializer`

Ce classe permet de définir les paramètres de Worker Serializer.

Les classes de model sont définis sous `/vitam/worker/common`.

8.2.4.2.16 Interface `HandlerIO`

Cet interface permet de définir les paramètres in et out de tous les Handlers.

Les classes de l'api sont définis sous `/vitam/worker/core/api`.

8.2.4.2.17 Classe `WorkerAction`

C'est l'interface contrat de tous les actions Handler event. Un action Handler doit implémenter cette interface .

Les classes de l'implémentation sont définis sous `/vitam/worker/core/impl`.

8.2.4.2.18 Classe `HandlerIOImpl`

Ce classe définit les paramètres in et out d'un Handler

How to use : Pour créer un Plugin :

- extends Abstract Class Action Handler
- implementer l'interface VitamAutoCloseable pour garantir qu'une ressource sera fermée lorsqu'elle n'est plus utilisée.
- Un constructeur par défaut
- **redéfinir la méthode execute de l'Action Handler :**
    - Paramètre WorkerParameters et Handler IO
    - type de retour Item Status
    - throws Processing Exception
- **faire l'override de méthode CheckMandatoryIOParameter**
    - Paramètre Handler IO
    - throws Processing Exception

8.2.5 Common

8.2.5.1 Présentation

8.2.5.2 Format Identifiers

Les services d'identification de formats peuvent être déployés sur tous les serveurs applicatifs vitam.

8.2.5.2.1 Configuration des services d'identification des formats

Dans **/vitam/conf** du serveur applicatif où sont déployés les services d'identification de formats, il faut un fichier **format-identifiers.conf**. C'est un fichier YAML de configuration des services d'identification de format. Il possède les configurations des services que l'on souhaite déployer sur le serveur.

Le code suivant contient un exemple de toutes les configurations possibles :

```
  siegfried-local:
type: SIEGFRIED
client: http
host: localhost
port: 55800
rootPath: /root/path
versionPath: /root/path/version/folder
createVersionPath: false
  mock:
type: MOCK
```

- **Le service Mock :**
  - identifié par *mock*
  - *type* : le type de service déployé : *MOCK*
- **Le service Siegfried :**
  - identifié par *siegfried-local*
  - *type* : le type de service déployé : *SIEGFRIED*
  - *client* : type de client (pour le moment seul *http* existe).
  - *host* : le host du serveur siegfried déployé (devrait être le host du serveur courant)
  - *port* : le port du serveur siegfried déployé
  - *rootPath* : la racine sur laquelle le service Siegfried doit résoudre les fichiers à tester (ex : « /data »)
  - *versionPath* : le chemin vers un dossier vide pour renvoyer la version (Doit posséder des droits en lecture)
  - *createVersionPath* : Si *false* le dossier doit pré-existant sur le server sur lequel tourne Siegfried. Sinon, le client siegfried tente de créer automatiquement le dossier en local.

NOTE : Chaque serveur est en charge de décrire la configuration nécessaire

8.2.6 Functional administration

8.2.6.1 Présentation

8.2.6.2 Configuration / fichiers utiles

Les fichiers de configuration sont gérés par les procédures d'installation ou de mise à niveau de l'environnement *VITAM*. Se référer au *DIN*.

Les fichiers de configuration sont définis sous `/vitam/conf/functional-administration`.

8.2.6.2.1 Fichier `functional-administration.conf`

ce fichier permet de définir l'URL d'accès au access server.

```
# Configuration MongoDB
mongoDbNodes:
{% for host in groups['hosts_mongos_data'] %}
- dbHost: {{ hostvars[host]['ip_service'] }}
  dbPort: {{ mongodb.mongos_port }}
{% endfor %}
dbName: masterdata
dbAuthentication: {{ mongodb.mongo_authentication }}
dbUserName: {{ mongodb['mongo-data'].functionalAdmin.user }}
dbPassword: {{ mongodb['mongo-data'].functionalAdmin.password }}

#Basic Authentication
adminBasicAuth:
- userName: {{ admin_basic_auth_user }}
  password: {{ admin_basic_auth_password }}

jettyConfig: jetty-config.xml
workspaceUrl: {{vitam.workspace | client_url}}
processingUrl: {{vitam.processing | client_url}}

# ElasticSearch
clusterName: {{ vitam_struct.cluster_name }}
elasticsearchNodes:
{% for host in groups['hosts_elasticsearch_data'] %}
- hostName: {{ hostvars[host]['ip_service'] }}
  tcpPort: {{ elasticsearch.data.port_tcp }}
{% endfor %}


# ExternalId configuration
listEnableExternalIdentifiers:
{% for tenant in vitam_tenants_usage_external %}
{% if tenant.identifiers is defined %}
  {{ tenant.name }}:
{% for external in tenant.identifiers %}
    - {{ external }}
{% endfor %}
{% endif %}
{% endfor %}

listMinimumRuleDuration:
{% for tenant in vitam_tenant_rule_duration %}
  {{ tenant.name }}:
{% for rule in tenant.rules %}
{% for key, value in rule.items() %}
    {{ key }}: {{ value }}
{% endfor %}
{% endfor %}
{% endfor %}
```

8.2.6.2.2 Passage des identifiants des référentiels en mode esclave

La génération des identifiants des référentiels est gérée par Vitam quand il fonctionne en mode maître.

Par exemple :

- Préfixé par `PR-` our les profils
- Préfixé par `IC-` pour les contrats d'entrée
- Préfixé par `AC-` pour les contrats d'accès

Si vous souhaitez gérer vous-même les identifiants sur un service référentiel, il faut qu'il soit en mode esclave.

Note : Cette modification de comportement est réalisable post-installation. Une interruption temporaire de service est à prévoir (redémarrage du service `vitam-functional-administration`.

Par défaut, tous les services référentiels de *VITAM* fonctionnent en mode maître. Pour désactiver le mode maître de Vitam, il faut modifier le fichier de configuration `/vitam/conf/functional-administration/functional-administration.conf`.

```
# ExternalId configuration

listEnableExternalIdentifiers:
 0:
   - INGEST_CONTRACT
   - ACCESS_CONTRACT
 1:
   - INGEST_CONTRACT
   - ACCESS_CONTRACT
   - PROFILE
   - SECURITY_PROFILE
   - CONTEXT
```

Depuis la version 1.0.4, la configuration par défaut de Vitam autorise des identifiants externes (ceux qui sont dans le fichier json importé).

- pour le tenant 0 pour les référentiels : contrat d'entrée et contrat d'accès.
- pour le tenant 1 pour les référentiels : contrat d'entrée, contrat d'accès, profil, profil de sécurité et contexte.

La liste des choix possibles, pour chaque tenant, est :

- INGEST_CONTRACT : contrats d'entrée
- ACCESS_CONTRACT : contrats d'accès
- PROFILE : profils SEDA
- SECURITY_PROFILE : profils de sécurité (utile seulement sur le tenant d'administration)
- CONTEXT : contextes applicatifs (utile seulement sur le tenant d'administration)
- ARCHIVE_UNIT_PROFILE : profils d'unités archivistiques

Note : se référer au métier pour ces choix.

Avertissement : Cette modification implique le redémarrage du/des composants (si mono-instance ou multi-instances du composant).

Prudence : En mode « esclave », il est fortement recommandé de faire débuter les référentiels avec d'autres chaînes de caractères que celle définies en mode « maître ».

Prudence : Ne pas oublier de répercuter cette modification sur le site secondaire

8.2.6.2.3 Configuration du Functional administration

functional-administration.conf : Fichier Yaml de configuration du server *worker*. Il possède une propriété :

- **listMinimumRuleDuration** : la durée minimum de chaque type de règle par tenant

```
listMinimumRuleDuration:
  2:
    AppraisalRule : 1 year
```

8.2.6.3 Opérations

- Démarrage du service

En tant qu'utilisateur root : `systemctl start vitam-functional-administration`

- Arrêt du service

En tant qu'utilisateur root : `systemctl stop vitam-functional-administration`

- Sauvegarde du service

Ce service ne nécessite pas de sauvegarde particulière.

- Supervision du service

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port>/functional-administration/v1/status`

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port admin>/admin/v1/status`

- Exports

N/A

- gestion de la capacité

N/A

- actions récurrentes

- cas des batches

N/A

8.2.7 Hello World Plugin

8.2.7.1 Présentation

Le composant hello-world-plugin est un exemple qui montre comment développer un plugin custom. Il suffit donc de prendre ce projet maven comme exemple et d'adapter le plugin à souhait.

Note : Sur le **pom** de ce module, seule la dépendance vers `common-plugin` est nécessaire.

```
<dependency>
    <groupId>fr.gouv.vitam</groupId>
    <artifactId>common-plugin</artifactId>
    <version>${vitam.version}</version>
</dependency>
```

Vous pouvez bien sûr ajouter d'autres dépendances qui servent à votre *plugin custom*.

8.2.7.1.1 Comment intégrer votre plugins dans vitam ?

Après avoir développé votre *plugin* en suivant les consignes ci-dessus, il faut faire ce qui suit :

- Générer votre jar
- Copier votre jar manuellement dans le dossier `/vitam/conf/worker/plugins-workspace/`, ou bien copier le dans le dossier de déploiement ansible `~/vitam/deployment/ansible-vitam/roles/vitam/files/worker/plugins-workspace/`
- Modifier le fichier plugins.json qui se trouve soit dans le dossier `/vitam/conf/worker/plugins.json` en production, ou bien dans le dossier de déploiement ansible qui se trouve :`vitam/deployment/ansible-vitam/roles/vitam/files/worker/plugins.json`, comme suit :

```
"HELLO_WORLD_PLUGIN": {
  "className": "fr.vitam.plugin.custom.HelloWorldPlugin",
  "propertiesFile": "hello_world_plugin.properties",
  "jarName": "hello-world-plugin-1.14.0-SNAPSHOT.jar"
}
```

Avertissement : jarName doit contenir uniquement le nom du jar avec extension ".jar"

A présent sur n'importe quel workflow, vous pouvez ajouter une action ayant comme « actionKey » la clé de votre plugin. Dans cet exemple : actionKey=HELLO_WORLD_PLUGIN

8.2.7.1.2 Créer un nouveau *workflow*

Tout d'abord création d'un nouveau workflow :

- Créer un nouveau *workflow* peut se résumer juste à copier un *workflow* existant et modifier son *identifier* et son *workerGroupId* pour, par exemple, utiliser des machines plus puissantes pour ce *workflow*
- L'identifier d'un *workflow* doit être UNIQUE, sinon un *workflow* existant portant le même *identifier* sera remplacé par le nouveau.
- La valeur de `typeProc` n'est pas actuellement dynamique (veuillez vous référer à l'enum **LogbookTypeProcess** pour voir les différentes valeurs possibles)

- La valeur `actionKey` doit être soit le *handler name* d'un *plugin* ou *handler* existant, soit celui d'un nouveau *plugin*.

Exemple d'un *workflow* :

```
{
  "id": "SampleIngestWorkflow",
  "name": "Sample Ingest Workflow",
  "identifier": "SAMPLE_PROCESS_SIP_UNITARY",
  "typeProc": "INGEST",
  "comment": "Sample Ingest Workflow V6",
  "steps": [
    {
      "workerGroupId": "DefaultWorker",
      "stepName": "STP_INGEST_CONTROL_SIP",
      "behavior": "BLOCKING",
      "distribution": {
        "kind": "REF"
      },
      "actions": [
        {
          "action": {
            "actionKey": "HELLO_WORLD_PLUGIN",
            "behavior": "BLOCKING",
            "in": [
              {
                "name": "var_name",
                "uri": "VALUE:Hello World"
              }
            ]
          }
        }
      ]
    }
  ]
}
```

Avertissement : Le fichier *workflow* doit être un fichier json avec comme extension (`.json`) sinon le fichier ne sera pas pris en compte.

8.2.7.1.2.1 Comment ajouter un nouveau workflow dans vitam ?

Il tout d'abord créer un fichier json avec un nom de votre choix et ayant la forme de l'exemple ci-dessus. Veuillez vous référer au différents workflow existants pour avoir plus d'information.

Il faut ensuite copier ce fichier (`CustomWorkflow.json`) dans :

- En production : Manuellement dans le dossier `/vitam/conf/processing/workflows/`
- Via ansible : Dans le dossier `~/vitam/deployment/ansible-vitam/roles/vitam/files/processing/workflows/`

8.2.7.1.2.2 Comment ajouter la traduction de clés des Plugins ?

On peut dans n'importe quel service vitam ajouter dans le dossier conf les deux fichiers suivants : - `vitam-logbook-messages_fr.properties` - `vitam-error-messages.properties`

Ces deux fichiers garantissent la traduction des clés.

Pour le nouveau plugins ajouté, le fichier properties qui est à l'intérieur du jar n'est visible que par le service (worker). Pour qu'on puisse avoir ces clés traduites,le fichier `vitam-logbook-messages_fr.properties` doit contenir les traductions des clés de ce nouveau *plugin*. Il faut copier ce fichier dans le dossier de conf de processing s'il n'existe pas.

```
HELLO_WORLD_PLUGIN=Test d''un plugin Hello World
HELLO_WORLD_PLUGIN.OK=Test d''un plugin Hello World réalisé avec succès
HELLO_WORLD_PLUGIN.KO=Échec lors du test d''un plugin Hello World
HELLO_WORLD_PLUGIN.FATAL=Erreur fatale lors du test d''un plugin Hello World
HELLO_WORLD_PLUGIN.WARNING=Avertissement lors du test d''un plugin Hello␣
↪World
```

8.2.7.1.2.3 Comment appeler le nouveau workflow ?

En utilisant l'API d'*ingest* et en passant les paramètres suivants :

- `X_CONTEXT_ID` : l'identifier de votre workflow (dans l'exemple ci-dessus SAMPLE_PROCESS_SIP_UNITARY)
- `X_ACTION` : votre action (RESUME, NEXT)

Le reste se fait automatiquement par le *back office*.

8.2.7.1.2.4 Remarques

- L'ajout d'un *workflow* dans processing en production ne nécessite pas de redémarrage. Un thread passe chaque heure configurale pour recharger les derniers workflow (ajoutés ou modifiés)
- L'ajout d'un jar dans les workers et les fichiers properties nécessitent, cependant, le redémarrage des workers et des services concernés.

8.2.7.1.2.5 Securité

Les plugins externes sont exécutés au même niveau de sécurité que ceux interne à *VITAM*. L'isolation de l'exécution des plugins externes n'est pas assurée par *VITAM*. C'est donc à l'exploitant de garantir la sécurité des plugins intégrés.

8.2.8 ihm-demo

8.2.8.1 Présentation

Cette IHM a été développée pour des fins de tests de VITAM.

Rôle :

- Permettre une utilisation basique de VITAM, notamment sans SIA

Fonctions :

- Représentation des arborescences et des graphes
- Formulaires dynamiques
- Suivi des opérations
- Gestion des référentiels

8.2.8.2 Configuration / fichiers utiles

Les fichiers de configuration sont gérés par les procédures d'installation ou de mise à niveau de l'environnement *VITAM*. Se référer au *DIN*.

Les fichiers de configuration sont définis sous `/vitam/conf/ihm-demo`.

8.2.8.2.1 Fichier `access-external-client.conf`

ce fichier permet de définir l'URL d'accès au access server.

```
serverHost: {{ vitam.accessexternal.host }}
serverPort: {{ vitam.accessexternal.port_service }}
secure: true
sslConfiguration :
 keystore :
  - keyPath: {{ vitam_folder_conf }}/keystore_{{ vitam_struct.vitam_component }}.p12
    keyPassword: {{ keystores.client_external.ihm_demo }}
 truststore :
  - keyPath: {{ vitam_folder_conf }}/truststore_{{ vitam_struct.vitam_component }}.jks
    keyPassword: {{ truststores.client_external }}
hostnameVerification: true
```

8.2.8.2.2 Fichier `ihm-demo.conf`

```
serverHost: {{ ip_service }}
port: {{ vitam_struct.port_service }}

baseUrl: "{{ vitam_struct.baseurl }}"
staticContent: {{ vitam_struct.static_content }}
baseUri: /{{ vitam_struct.baseuri }}
secureMode:
{% for realm in vitam_struct.authentication_realms %}
- {{ realm }}
{% endfor %}

jettyConfig: jetty-config.xml
authentication: true
enableXsrFilter: true
enableSession: true

allowedMediaTypes:
{% for mediaType in vitam_struct.allowedMediaTypes %}
    - type: {{ mediaType.type }}
      subtype: {{ mediaType.subtype }}
{% endfor %}
```

tenants : liste des tenants disponibles sur l'ihm-demo.

8.2.8.2.3 Fichier `ingest-external-client.conf`

```
serverHost: {{ vitam.ingestexternal.host }}
serverPort: {{ vitam.ingestexternal.port_service }}
secure: true
sslConfiguration :
 keystore :
  - keyPath: {{ vitam_folder_conf }}/keystore_{{ vitam_struct.vitam_component }}.p12
    keyPassword: {{ keystores.client_external.ihm_demo }}
 truststore :
  - keyPath: {{ vitam_folder_conf }}/truststore_{{ vitam_struct.vitam_component }}.jks
    keyPassword: {{ truststores.client_external }}
hostnameVerification: true
```

8.2.8.2.4 Fichier `shiro.ini`

```
# ======================
# Shiro INI configuration
# ======================

[main]
# Objects and their properties are defined here,
# Such as the securityManager, Realms and anything
# else needed to build the SecurityManager


# Cache Manager
builtInCacheManager = org.apache.shiro.cache.MemoryConstrainedCacheManager

# Security Manager
securityManager.cacheManager = $builtInCacheManager

sessionManager = org.apache.shiro.web.session.mgt.DefaultWebSessionManager
securityManager.sessionManager = $sessionManager
securityManager.sessionMode = native
securityManager.sessionManager.globalSessionTimeout = {{ vitam_struct.session_timeout␣
↪}}
securityManager.sessionManager.sessionIdUrlRewritingEnabled = false
securityManager.sessionManager.sessionIdCookie.secure = {{ vitam_struct.secure_cookie␣
↪}}
securityManager.rememberMeManager.cookie.secure = {{ vitam_struct.secure_cookie }}
securityManager.rememberMeManager.cookie.httpOnly = true

# Notice how we didn't define the class for the FormAuthenticationFilter ('authc') -␣
↪it is instantiated and available already:
authc.loginUrl = /#!/login


# credentialsMatcher
sha256Matcher = org.apache.shiro.authc.credential.Sha256CredentialsMatcher

{% if "iniRealm" in vitam_struct.authentication_realms %}
iniRealm.credentialsMatcher = $sha256Matcher
{% endif %}

{% if "ldapRealm" in vitam_struct.authentication_realms %}
contextFactory = org.apache.shiro.realm.ldap.JndiLdapContextFactory
```

(suite sur la page suivante)

(suite de la page précédente)

```
contextFactory.url = {{ ldap_authentification.ldap_protocol }}://{{ ldap_
↪authentification.ldap_server }}:{{ ldap_authentification.ldap_port }}
{% if ldap_authentification.ldap_login is defined and ldap_authentification.ldap_pwd␣
↪is defined %}
{% if ldap_authentification.ldap_login != "" and ldap_authentification.ldap_pwd != ""
↪%}
contextFactory.systemUsername = {{ ldap_authentification.ldap_login }}
contextFactory.systemPassword = {{ ldap_authentification.ldap_pwd }}
{% endif %}
{% endif %}

ldapRealm = fr.gouv.vitam.common.auth.core.realm.LdapRealm
ldapRealm.ldapContextFactory = $contextFactory
ldapRealm.searchBase = "{{ ldap_authentification.ldap_base }}"
ldapRealm.groupRequestFilter = {{ ldap_authentification.ldap_group_request }}
ldapRealm.userDnTemplate = {{ ldap_authentification.ldap_userDn_Template }}
ldapRealm.groupRolesMap = "{{ ldap_authentification.ldap_admin_group }}":"admin", "{{␣
↪ldap_authentification.ldap_user_group }}":"user", "{{ ldap_authentification.ldap_
↪guest_group }}":"guest"
{% endif %}

x509 = fr.gouv.vitam.common.auth.web.filter.X509AuthenticationFilter

x509.useHeader = False

x509credentialsMatcher = fr.gouv.vitam.common.auth.core.authc.
↪X509CredentialsSha256Matcher

{% if "x509Realm" in vitam_struct.authentication_realms %}
x509Realm = fr.gouv.vitam.common.auth.core.realm.X509KeystoreFileWithRoleRealm
x509Realm.grantedKeyStoreName = {{ vitam_folder_conf }}/grantedstore_ihm-demo.jks
x509Realm.grantedKeyStorePassphrase = {{ password_grantedstore }}
x509Realm.trustedKeyStoreName = {{ vitam_folder_conf }}/truststore_ihm-demo.jks
x509Realm.trustedKeyStorePassphrase = {{ password_truststore }}
x509Realm.credentialsMatcher = $x509credentialsMatcher
x509Realm.certificateDnRoleMapping = "CN=userAdmin,O=Vitam,L=Paris":"admin",
↪"CN=userUser,O=Vitam,L=Paris,C=FR":"user"
{% endif %}

securityManager.realms = {% for realm in vitam_struct.authentication_realms %}{% if␣
↪not loop.first %},{% endif %}${{ realm }}{% endfor %}

{% if "iniRealm" in vitam_struct.authentication_realms %}

[users]
# The 'users' section is for simple deployments
# # when you only need a small number of statically-defined
# # set of User accounts.
# #username = password
{% for item in vitam_users %}
 {{ item.login }}={{ item.password|hash('sha256') }}, {{ item.role }}
{% endfor %}

{% endif %}


[roles]
```

(suite sur la page suivante)

(suite de la page précédente)

```
admin = *
user = messages:*, archivesearch:*, logbook:*, ingest:*, archiveupdate:*,␣
↪archiveunit:*, ingests:read, admin:formats:read, admin:rules:read, admin:accession-
↪register:read, logbookunitlifecycles:*, logbookobjectslifecycles:*, clear:delete,␣
↪check:read, traceability:content:read, accesscontracts:read, profiles:read,␣
↪contracts:read, contexts:read, archiveunitprofiles:read, ontologies:read,␣
↪accessionregisterssymbolic:read
guest = archivesearch:*, archiveunit:*, units:*, unit:*, admin:accession-
↪register:read, accesscontracts:read


[urls]
# make sure the end-user is authenticated.  If not, redirect to the 'authc.loginUrl'␣
↪above,
# and after successful authentication, redirect them back to the original account␣
↪page they
# were trying to view:
/v1/api/login = anon
/v1/api/logout = logout
/v1/api/messages/logbook = anon
/v1/api/tenants = anon
/v1/api/securemode = anon
/v1/api/admintenant = anon
/v1/api/permissions = x509
/v1/api/** = authc, x509
/#/** = authc
```

8.2.8.3 Configuration de apache shiro

8.2.8.3.1 Présentation authentification via LDAP et via certificat

Afin de pouvoir authentifier des clients via une base de données LDAP il suffit de bien configurer shiro. Pour ce faire, Vitam utilise le fichier shiro.ini qui a la forme suivante.

```
[main]
contextFactory = org.apache.shiro.realm.ldap.JndiLdapContextFactory
contextFactory.url = ldap://localhost:389
contextFactory.systemUsername = cn=admin,dc=example,dc=org
contextFactory.systemPassword = password
realm = fr.gouv.vitam.common.security.rest.LdapRealm
realm.ldapContextFactory = $contextFactory
realm.searchBase = "dc=example,dc=org"
realm.groupRequestFilter = (&(objectClass=groupOfNames)(member={0}))
realm.userDnTemplate = uid={0},dc=example,dc=org
realm.groupRolesMap = "cn=gadmins,dc=example,dc=org":"admin", "cn=gusers,dc=example,
↪dc=org":"user", "cn=gadmins,dc=example,dc=org":"guest"
securityManager.realms = $realm
```

```
x509 = fr.gouv.vitam.common.auth.web.filter.X509AuthenticationFilter
x509.useHeader = false
x509credentialsMatcher = fr.gouv.vitam.common.auth.core.authc.
↪X509CredentialsSha256Matcher
x509Realm = fr.gouv.vitam.common.auth.core.realm.X509KeystoreFileWithRoleRealm
x509Realm.grantedKeyStoreName = /vitam/conf/ihm-demo/grantedstore_ihm-demo.jks
```

(suite sur la page suivante)

(suite de la page précédente)

```
x509Realm.grantedKeyStorePassphrase = azerty12
x509Realm.trustedKeyStoreName = /vitam/conf/ihm-demo/truststore_ihm-demo.jks
x509Realm.trustedKeyStorePassphrase = azerty10
x509Realm.credentialsMatcher = $x509credentialsMatcher
x509Realm.certificateDnRoleMapping = "CN=userAdmin,O=Vitam,L=Paris":"admin",
↪"CN=userUser,O=Vitam,L=Paris,C=FR":"user"
securityManager.realms = $x509Realm
```

8.2.8.3.2 Décryptage de `shiro.ini`

`[main]` Contient la déclaration des options et mappings dans l'authentication ldap :

- contextFactory.url : url du serveur ldap ;
- contextFactory.systemUsername : identifiant de l'utilisateur ;
- contextFactory.systemPassword : mot de passe ;
- realm.searchBase : le domaine de recherche dans LDAP ;
- realm.groupRequestFilter : chaque utilisateur est déclaré dans un groupe, cette requête sert à chercher les groupes de l'utilisateur ;
- realm.userDnTemplate : le modèle pour traduire un identifiant de l'utilisateur en DN (distinguished name) dans ldap ;
- realm.groupRolesMap : le mapping entre le DN des group de l'utilisateur et les rôles dans ihm ;
- x509Realm.grantedKeyStoreName : le fichier grantedstore ;
- x509Realm.trustedKeyStoreName : le fichier trustedstore ;
- x509Realm.certificateDnRoleMapping : le mapping entre le DisplayName de certificat et les rôles dans ihm.

Note : on peut déclarer plusieurs groups qui ont la même rôle admin avec cette syntaxe :

```
"groupeA" : "admin", "groupeB" : "admin", "groupeC" : "admin"
```

8.2.8.4 Opérations

- Démarrage du service

En tant qu'utilisateur root : `systemctl start vitam-ihm-demo`

- Arrêt du service

En tant qu'utilisateur root : `systemctl stop vitam-ihm-demo`

- Sauvegarde du service

Ce service ne nécessite pas de sauvegarde particulière.

- Supervision du service

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port>/ihm-demo/v1/status`

- Gestion des utilisateurs

Les utilisateurs sont actuellement gérés via le fichier `shiro.ini`, dans la section `[users]`.

- Créer un utilisateur

  Lancer la commande shell suivante pour générer le mot de passe :

```
echo -n <motdepasse> | sha256sum
```

Copier le résultat.
Ensuite, éditer le fichier `/vitam/conf/ihm-demo/shiro.ini` et ajouter, dans la section `[users]`, la ligne suivante :

```
<login de l'utilisateur>=<résultat de la commande de génération de mot de␣
↪passe précédente>
```

Pour terminer, relancer le service `vitam-ihm-demo` par la commande :

```
systemctl restart vitam-ihm-demo
```

- Supprimer un utilisateur

Dans la section `[users]`, enlever la ligne correspondant à l'utilisateur à supprimer. Pour terminer, relancer le service `vitam-ihm-demo` par la commande :

```
systemctl restart vitam-ihm-demo
```

- Exports

N/A

- gestion de la capacité

N/A

- actions récurrentes
- cas des batches

N/A

8.2.9 ihm-recette

8.2.9.1 Présentation

Cette IHM a été développée pour des fins de validation de VITAM. Elle permet de réaliser des tests de non-régression, mais également des actions sur le contenu des bases de données.

Danger : Cette IHM ne doit PAS être déployée dans un environnement de production !

8.2.9.2 Configuration / fichiers utiles

Les fichiers de configuration sont gérés par les procédures d'installation ou de mise à niveau de l'environnement *VITAM*. Se référer au *DIN*.

Les fichiers de configuration sont définis sous `/vitam/conf/ihm-recette`.

8.2.9.2.1 Fichier `access-external-client.conf`

ce fichier permet de définir l'URL d'accès au access server.

```
serverHost: {{ vitam.accessexternal.host }}
serverPort: {{ vitam.accessexternal.port_service }}
secure: true
sslConfiguration :
 keystore :
  - keyPath: {{ vitam_folder_conf }}/keystore_{{ vitam_struct.vitam_component }}.p12
    keyPassword: {{ keystores.client_external.ihm_recette }}
 truststore :
  - keyPath: {{ vitam_folder_conf }}/truststore_{{ vitam_struct.vitam_component }}.jks
    keyPassword: {{ truststores.client_external }}
hostnameVerification: false
```

8.2.9.2.2 Fichier `driver-location.conf`

```
driverLocation: {{ vitam_folder_lib }}
```

8.2.9.2.3 Fichier `driver-mapping.conf`

```
driverMappingPath: {{ vitam_folder_data }}/
delimiter: ;
```

8.2.9.2.4 Fichier `functional-administration-client.conf`

```
serverHost: {{ vitam.functional_administration.host }}
serverPort: {{ vitam.functional_administration.port_service }}
```

8.2.9.2.5 Fichier `ihm-recette-client.conf`

```
serverHost: {{ vitam_struct.host }}
serverPort: {{ vitam_struct.port_service }}
```

8.2.9.2.6 Fichier `ihm-recette.conf`

```
serverHost: {{ ip_service }}
port: {{ vitam_struct.port_service }}

baseUrl: "/{{ vitam_struct.baseuri }}"
baseUri: "/{{ vitam_struct.baseuri }}"
staticContent: "{{ vitam_struct.static_content }}"

jettyConfig: jetty-config.xml
authentication: true
enableXsrFilter: true
enableSession: true

secureMode:
```

(suite sur la page suivante)

(suite de la page précédente)

```
{% for securemode in vitam_struct.secure_mode %}
- {{ securemode }}
{% endfor %}
sipDirectory: {{ vitam_folder_data }}/test-data
performanceReportDirectory: {{ vitam_folder_data }}/report/performance

testSystemSipDirectory: {{ vitam_folder_data }}/test-data/system
testSystemReportDirectory: {{ vitam_folder_data }}/report/system
ingestMaxThread: {{ ansible_processor_cores * ansible_processor_threads_per_core + 1 }
↪}

#
workspaceUrl: {{vitam.workspace | client_url}}

# Configuration MongoDB
mongoDbNodes:
{% for server in groups['hosts_mongos_data'] %}
- dbHost: {{ hostvars[server]['ip_service'] }}
  dbPort: {{ mongodb.mongos_port }}
{% endfor %}
# Actually need this field for compatibility
dbName: admin
# @integ: parametrize it !
masterdataDbName: masterdata
logbookDbName: logbook
metadataDbName: metadata
dbAuthentication: {{ mongodb.mongo_authentication }}
dbUserName: {{ mongodb['mongo-data']['admin']['user'] }}
dbPassword: {{ mongodb['mongo-data']['admin']['password'] }}

# ElasticSearch
clusterName: {{ vitam_struct.cluster_name }}
elasticsearchNodes:
{% for server in groups['hosts_elasticsearch_data'] %}
- hostName: {{ hostvars[server]['ip_service'] }}
  tcpPort: {{ elasticsearch.data.port_tcp }}
{% endfor %}
```

8.2.9.2.7 Fichier `ingest-external-client.conf`

```
serverHost: {{ vitam.ingestexternal.host }}
serverPort: {{ vitam.ingestexternal.port_service }}
secure: true
sslConfiguration :
 keystore :
  - keyPath: {{ vitam_folder_conf }}/keystore_{{ vitam_struct.vitam_component }}.p12
    keyPassword: {{ keystores.client_external.ihm_recette }}
 truststore :
  - keyPath: {{ vitam_folder_conf }}/truststore_{{ vitam_struct.vitam_component }}.jks
    keyPassword: {{ truststores.client_external }}
hostnameVerification: false
```

8.2.9.2.8 Fichier `shiro.ini`

```
[main]

{% if vitam_struct.secure_mode == 'x509' %}
x509 = fr.gouv.vitam.common.auth.web.filter.X509AuthenticationFilter

x509.useHeader = {{ vitam_ssl_user_header }}

x509credentialsMatcher = fr.gouv.vitam.common.auth.core.authc.
↪X509CredentialsSha256Matcher

x509Realm = fr.gouv.vitam.common.auth.core.realm.X509KeystoreFileRealm
x509Realm.grantedKeyStoreName = {{ vitam_folder_conf }}/grantedstore_ihm-recette.jks
x509Realm.grantedKeyStorePassphrase = {{ password_grantedstore }}
x509Realm.trustedKeyStoreName = {{ vitam_folder_conf }}/truststore_ihm-recette.jks
x509Realm.trustedKeyStorePassphrase = {{ password_truststore }}
x509Realm.credentialsMatcher = $x509credentialsMatcher
securityManager.realm = $x509Realm
securityManager.subjectDAO.sessionStorageEvaluator.sessionStorageEnabled = false
[urls]
/v1/api/** = x509

{% else %}
# Objects and their properties are defined here,
# Such as the securityManager, Realms and anything
# else needed to build the SecurityManager
# credentialsMatcher
sha256Matcher = org.apache.shiro.authc.credential.Sha256CredentialsMatcher
iniRealm.credentialsMatcher = $sha256Matcher
# Cache Manager
builtInCacheManager = org.apache.shiro.cache.MemoryConstrainedCacheManager
# Security Manager
securityManager.cacheManager = $builtInCacheManager
sessionManager = org.apache.shiro.web.session.mgt.DefaultWebSessionManager
securityManager.sessionManager = $sessionManager
securityManager.sessionMode=native
securityManager.sessionManager.globalSessionTimeout = {{ vitam_struct.session_timeout␣
↪}}
securityManager.sessionManager.sessionIdUrlRewritingEnabled = false
securityManager.sessionManager.sessionIdCookie.secure = {{ vitam_struct.secure_cookie␣
↪}}
securityManager.rememberMeManager.cookie.secure = {{ vitam_struct.secure_cookie }}
securityManager.rememberMeManager.cookie.httpOnly = true
# Notice how we didn't define the class for the FormAuthenticationFilter ('authc') -␣
↪it is instantiated and available already:
authc.loginUrl = /#!/login
[users]
# The 'users' section is for simple deployments
# when you only need a small number of statically-defined
# set of User accounts.
#username = password
{% for item in vitam_users %}
{% if item.role == "admin" %}
{{ item.login }}={{ item.password|hash('sha256') }}
{% endif %}
{% endfor %}
```

(suite sur la page suivante)

(suite de la page précédente)

```
[roles]
# The 'roles' section is for simple deployments
# when you only need a small number of statically-defined
# roles.
[urls]
# make sure the end-user is authenticated.  If not, redirect to the 'authc.loginUrl'␣
↪above,
# and after successful authentication, redirect them back to the original account␣
↪page they
# were trying to view:
/v1/api/login = anon
/v1/api/logout = logout
/v1/api/securemode = anon
/** = authc

{% endif %}
```

8.2.9.2.9 Fichier `static-offer.json`

```
{% if vitam.storageofferdefault.https_enabled==true %}
    {% set protocol = 'https' %}
{% else %}
    {% set protocol = 'http' %}
{% endif %}
[
{% for item in vitam_strategy %}
{
    "id" : "{{ item.name }}.service.{{ item.vitam_site_name |default(vitam_site_name)␣
↪}}.{{ consul_domain }}",
    "baseUrl" : "{{ protocol }}://{{ item.name }}.service.{{ item.vitam_site_name␣
↪|default(vitam_site_name) }}.{{ consul_domain }}:{{ vitam.storageofferdefault.port_
↪service }}",
    {% if item.asyncRead is defined %} "asyncRead": {{item.asyncRead|lower }}, {%␣
↪endif %}
    "parameters" : {
        {% if vitam.storageofferdefault.https_enabled==true %}
        "keyStore-keyPath": "{{ vitam_folder_conf }}/keystore_storage.p12",
        "keyStore-keyPassword": "{{ keystores.client_storage.storage }}",
        "trustStore-keyPath": "{{ vitam_folder_conf }}/truststore_storage.jks",
        "trustStore-keyPassword": "{{ truststores.client_storage }}"
        {% endif %}
    }
}
{% if not loop.last %},
{% endif %}
{% endfor %}
]
```

8.2.9.2.10 Fichier `static-strategy.json`

```
[
    {
```

(suite sur la page suivante)

(suite de la page précédente)

```
        "id" : "default",
        "offers" : [
            {% for item in vitam_strategy %}
            {"id" : "{{ item.name }}.service.{{ item.vitam_site_name |default(vitam_
↪site_name) }}.{{ consul_domain }}"{% if item.referent is defined %}{% if item.
↪referent|lower == "true" %}, "referent" : true{% endif %}{% endif %}{% if item.
↪status is defined %}, "status" :  "{{ item.status| upper }}" {% endif %}}{% if not␣
↪loop.last %},{% endif %}
            {% endfor %}
        ]
    }
    {% if other_strategies is defined %}
    {% for strategy_name, strategy_offers in other_strategies.iteritems() %}
    ,
    {
        "id" : "{{ strategy_name }}",
        "offers" : [
            {% for strategy_offer in strategy_offers %}
            {"id" : "{{ strategy_offer.name }}.service.{{ strategy_offer.vitam_site_
↪name |default(vitam_site_name) }}.{{ consul_domain }}"{% if strategy_offer.status␣
↪is defined %}, "status" :  "{{ strategy_offer.status| upper }}" {% endif %}}{% if␣
↪not loop.last %},{% endif %}
            {% endfor %}
        ]
    }
    {% endfor %}
    {% endif %}

]
```

8.2.9.2.11 Fichier `storage-client.conf`

```
serverHost: {{ vitam.storageengine.host }}
serverPort: {{ vitam.storageengine.port_service }}
```

8.2.9.2.12 Fichier `storage.conf`

```
urlWorkspace: {{ vitam.workspace | client_url }}
timeoutMsPerKB: 100
jettyConfig: jetty-config.xml
zippingDirecorty: {{ vitam_folder_data }}/storage_archives
loggingDirectory: {{ vitam_folder_log }}
```

8.2.9.2.13 Fichier `storage-offer.conf`

```
strategy_name=[{% for item in vitam_strategy %}"{{ item.name }}.service.{{ consul_
↪domain }}"{% if not loop.last %},{% endif %}{% endfor %}]
```

8.2.9.2.14 Fichier `tnr.conf`

```
urlWorkspace: {{vitam.workspace | client_url}}
tenantsTest: [ "0" ]
vitamSecret: {{ plateforme_secret }}
tenants: [ "{{ vitam_tenant_ids | join('", "') }}" ]
adminTenant: {{ vitam_tenant_admin }}
```

8.2.9.3 Opérations

- Démarrage du service

En tant qu'utilisateur root : `systemctl start vitam-ihm-recette`

- Arrêt du service

En tant qu'utilisateur root : `systemctl stop vitam-ihm-recette`

- Sauvegarde du service

Ce service ne nécessite pas de sauvegarde particulière.

- Supervision du service

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port admin>/admin/v1/status`

- Gestion des utilisateurs

Les utilisateurs sont actuellement gérés via le fichier `shiro.ini`, dans la section `[users]`.

- Créer un utilisateur

Lancer la commande shell suivante pour générer le mot de passe :

```
echo -n <motdepasse> | sha256sum
```

Copier le résultat.
Ensuite, éditer le fichier `/vitam/conf/ihm-recette/shiro.ini` et ajouter, dans la section `[users]`, la ligne suivante :

```
<login de l'utilisateur>=<résultat de la commande de génération de mot de␣
↪passe précédente>
```

Pour terminer, relancer le service `vitam-ihm-recette` par la commande :

```
systemctl restart vitam-ihm-recette
```

- Supprimer un utilisateur

Dans la section `[users]`, enlever la ligne correspondant à l'utilisateur à supprimer. Pour terminer, relancer le service `vitam-ihm-recette` par la commande :

```
systemctl restart vitam-ihm-recette
```

- Exports

N/A

- gestion de la capacité

N/A

- actions récurrentes

- cas des batches

N/A

8.2.10 Ingest

8.2.10.1 Introduction

Ce document présente les configurations pour utiliser les différents modules de *ingest*.

8.2.10.2 ingest-external

8.2.10.2.1 Présentation

Ingest-external est le composant d'interface entre *VITAM* et un *SIA* client, permettant de réaliser des entrées d'archives dans *VITAM*.

Rôle :

- Exposer les API publiques du système
- Sécuriser l'accès aux API de VITAM

8.2.10.2.2 Configuration / fichiers utiles

Les fichiers de configuration sont gérés par les procédures d'installation ou de mise à niveau de l'environnement *VITAM*. Se référer au *DIN*.

Les fichiers de configuration sont définis sous `/vitam/conf/ingest-external`.

8.2.10.2.2.1 Fichier `ingest-external.conf`

```
path: {{ vitam_folder_data }}
jettyConfig: jetty-config.xml
authentication: false
tenantFilter : true
antiVirusScriptName: scan-{{ vitam_struct.antivirus }}.sh
timeoutScanDelay: {{ vitam_struct.scantimeout|default(60000) }}
baseUploadPath: {{ vitam_struct.upload_dir }}
successfulUploadDir: {{ vitam_struct.success_dir }}
failedUploadDir: {{ vitam_struct.fail_dir }}
fileActionAfterUpload: {{ vitam_struct.upload_final_action}}
```

Ce fichier contient un appel au shell d'antivirus (par défaut, ClamAV) ; se reporter au *DIN*.

Il est possible, dans le cas de fichiers SIP volumineux, d'héberger des fichiers directement dans ingest-external (valeur de la directive `baseUploadPath`). Ces fichiers doivent être accessibles et utilisables par le *user* système vitam.

Les options associées à cette fonctionnalité peuvent être paramétrées dans le fichier deployment/environment/group_vars/all/vitam_vars.yml avant installation de Vitam.

La directive `fileActionAfterUpload` accepte les valeurs :

- `NONE` : le fichier reste
- `MOVE` : déplace le fichiers vers les valeurs des directives `successfulUploadDir` (en cas de succès de l'ingest) et `failedUploadDir` (en cas de non-succès de l'ingest)
- `DELETE` : le fichier est supprimé en cas de succès de l'ingest uniquement

A charge à l'exploitant de bien gérer l'espace disque de ces répertoires (il faut penser aux ingests en échecs par exemple).

Se reporter au manuel de développement pour l'appel d'API associé.

8.2.10.2.2.2 Fichier `ingest-internal-client.conf`

```
serverHost: {{ vitam.ingestinternal.host }}
serverPort: {{ vitam.ingestinternal.port_service }}
```

8.2.10.2.2.3 Fichier `internal-security-client.conf`

```
serverHost: {{ vitam.security_internal.host }}
serverPort: {{ vitam.security_internal.port_service }}
secure: {{ vitam.security_internal.https_enabled }}
```

8.2.10.2.2.4 Fichier `format-identifiers.conf`

```
siegfried-local:
  type: SIEGFRIED
  client: http
  host: localhost
  port: {{ siegfried.port }}
  rootPath: {{ vitam_folder_data }}/
  versionPath: {{ vitam_folder_data }}/version/folder
```

8.2.10.2.2.5 Fichier `functional-administration-client.conf`

```
serverHost: {{ vitam.functional_administration.host }}
serverPort: {{ vitam.functional_administration.port_service }}
```

8.2.10.2.2.6 Fichier `scan-clamav.sh`

Ce script de *scan* appelle l'antivirus (par défaut, clamAV ; ce paramètre est surchargeable à l'installation ; se référer au :term'DIN' pour plus de précisions) pour détecter les virus.

```
#!/bin/sh

#################################################################################
# Role:                                                                         #
# Scan un single file using clamav anti-virus                                   #
#################################################################################
# Args:                                                                         #
# - file to scan                                                                #
#################################################################################
# Return:                                                                       #
# - 0: scan OK - no virus                                                       #
RET_NOTVIRUS=0
# - 1: virus found and corrected                                                #
RET_VIRUS_FOUND_FIXED=1
# - 2: virus found but not corrected                                            #
RET_VIRUS_FOUND_NOTFIXED=2
# - 3: Fatal scan not performed                                                 #
```

(suite sur la page suivante)

(suite de la page précédente)

```
RET_FAILURE=3
# stdout : names of virus found (1 per line) if virus found ;             #
#          failure description if failure                                 #
# stderr : full ouput of clamav                                           #
###########################################################################

# Default return code : scan NOK
RET=3
OUTPUT_DIR=$(mktemp -d)
if [ $# -ne 1 ]; then # Argument number must be one
        echo "ERROR : $# parameter(s) provided, only one parameter is needed"
else # one argument, let's go
        if [ ! -f  "$1" ];then # if the file wich will be scan is existing, keep going
                echo "ERROR : \"$1\" doesn't exit"
        else
                clamdscan -z --config-file=/etc/clamd.d/scan.conf "$1" 1> ${OUTPUT_
↪DIR}/stdout 2> ${OUTPUT_DIR}/stderr # scanning the file and store the output OUTPUT
                RET=$? # return code of clamscan

                # Always output clamscan outputs to our own stderr
                (>&2 cat ${OUTPUT_DIR}/stdout  ${OUTPUT_DIR}/stderr)

                if [ ${RET} -eq ${RET_VIRUS_FOUND_FIXED} ] ; then
                        RET=2 # if virus found clamscan return 1; the script must␣
↪return 2
                        (>&1 cat ${OUTPUT_DIR}/stdout  | grep `basename ${1}` | cut -
↪d ' ' -f 2) # sending the list of virus to our own stdout
                elif [ ${RET} -eq 2 ] ; then
                        RET=3 # if scan not performed clamscan return 2; the script␣
↪must return 3
                        (>&1 cat ${OUTPUT_DIR}/stdout  | grep `basename ${1}` | cut -
↪d ' ' -f 2-) # sending the failure reason to our own stdout
                fi

                if [ -f  "${OUTPUT_DIR}/stdout" ]
                then
                        rm ${OUTPUT_DIR}/stdout
                fi
                if [ -f  "${OUTPUT_DIR}/stderr" ]
                then
                        rm ${OUTPUT_DIR}/stderr
                fi
        fi
fi
rmdir ${OUTPUT_DIR}
exit ${RET}
```

8.2.10.2.3 Opérations

- Démarrage du service

En tant qu'utilisateur root : `systemctl start vitam-ingest-external`

- Arrêt du service

En tant qu'utilisateur root : `systemctl stop vitam-ingest-external`

- Sauvegarde du service

Ce service ne nécessite pas de sauvegarde particulière.

- Supervision du service

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port>/ingest-external/v1/status`

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port admin>/admin/v1/status`

- Exports

N/A

- gestion de la capacité

N/A

- actions récurrentes

- cas des batches

N/A

8.2.10.3 ingest-internal

8.2.10.3.1 Présentation

Rôle :

- Permettre l'entrée d'une archive SEDA dans le SAE

Fonctions :

- Upload HTTP de fichiers au format SEDA
- Sas de validation antivirus des fichiers entrants
- Persistance du SEDA dans workspace
- Lancement des workflows de traitements liés à l'entrée dans processing

8.2.10.3.2 Configuration / fichiers utiles

Les fichiers de configuration sont gérés par les procédures d'installation ou de mise à niveau de l'environnement *VITAM*. Se référer au *DIN*.

Les fichiers de configuration sont définis sous `/vitam/conf/ingest-internal`.

8.2.10.3.2.1 Fichier `ingest-internal.conf`

```
workspaceUrl: {{vitam.workspace | client_url}}
processingUrl: {{vitam.processing | client_url}}
jettyConfig: jetty-config.xml
```

Ce fichier précise les URLs pour les services « Processing » et « Workspace », et la configuration du serveur jetty.

8.2.10.3.2.2 Fichier `storage-client.conf`

```
serverHost: {{ vitam.storageengine.host }}
serverPort: {{ vitam.storageengine.port_service }}
```

8.2.10.3.3 Opérations

- Démarrage du service

En tant qu'utilisateur root : `systemctl start vitam-ingest-internal`

- Arrêt du service

En tant qu'utilisateur root : `systemctl stop vitam-ingest-internal`

- Sauvegarde du service

Ce service ne nécessite pas de sauvegarde particulière.

- Supervision du service

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port>/ingest-internal/v1/status`

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port admin>/admin/v1/status`

- Exports

N/A

- gestion de la capacité

N/A

- actions récurrentes

- cas des batches

N/A

8.2.11 Security-Internal

8.2.11.1 Introduction

Ce document présente la configuration pour le module security-internal.

8.2.11.2 security-internal-exploitation

Ce document spécifie la configuration (fichiers de config) pour lancer le services de `security-internal`.

8.2.11.2.1 Fichier `security-internal.conf`

Ce fichier permet de définir la configuration du serveur MongoDB, du serveur jetty, les tenants, ainsi que la configuration de l'authentification *personae* pour les permissions des *endpoints* externes de *VITAM*.

```
# Configuration MongoDB
mongoDbNodes:
{% for host in groups['hosts_mongos_data'] %}
- dbHost: {{ hostvars[host]['ip_service'] }}
  dbPort: {{ mongodb.mongos_port }}
{% endfor %}
dbName: identity
dbAuthentication: {{ mongodb.mongo_authentication }}
dbUserName: {{ mongodb['mongo-data'].securityInternal.user }}
dbPassword: {{ mongodb['mongo-data'].securityInternal.password }}

jettyConfig: jetty-config.xml

personalCertificatePermissionConfig: personal-certificate-permissions.conf

#Basic Authentication
adminBasicAuth:
- userName: {{ admin_basic_auth_user }}
  password: {{ admin_basic_auth_password }}
```

8.2.11.2.2 Fichier `personal-certificate-permissions.conf`

Configuration des permissions nécessitant une authentification personae ou ne nécessitant pas d'authentification personae.

```
# Personal certification configuration for endpoint permissions

permissionsRequiringPersonalCertificate:



permissionsWithoutPersonalCertificate:
  - 'dipexport:create'
  - 'dipexportv2:create'
  - 'dipexport:id:dip:read'
  - 'transfers:create'
  - 'transfers:reply'
  - 'transfers:id:sip:read'
  - 'logbookobjectslifecycles:id:read'
  - 'logbookoperations:read'
  - 'logbookoperations:id:read'
  - 'logbookunitlifecycles:id:read'
  - 'units:read'
  - 'units:id:read:json'
  - 'units:id:update'
  - 'units:id:objects:read:json'
  - 'units:id:objects:read:binary'
  - 'units:update'
  - 'unitsWithInheritedRules:read'
  - 'units:rules:update'
  - 'accesscontracts:create:json'
  - 'accesscontracts:read'
  - 'accesscontracts:id:read'
  - 'accesscontracts:id:update'
  - 'accessionregisters:read'
```

(suite sur la page suivante)

(suite de la page précédente)

```
  - 'accessionregisters:id:accessionregisterdetails:read'
  - 'agencies:create'
  - 'agencies:read'
  - 'agencies:id:read'
  - 'agenciesfile:check'
  - 'agenciesreferential:id:read'
  - 'audits:create'
  - 'contexts:create:json'
  - 'contexts:read'
  - 'contexts:id:read'
  - 'contexts:id:update'
  - 'distributionreport:id:read'
  - 'formats:read'
  - 'formats:create'
  - 'formats:id:read'
  - 'formatsfile:check'
  - 'ingestcontracts:create:json'
  - 'ingestcontracts:read'
  - 'ingestcontracts:id:read'
  - 'ingestcontracts:id:update'
  - 'operations:read'
  - 'operations:id:read:status'
  - 'operations:id:read'
  - 'operations:id:update'
  - 'operations:id:delete'
  - 'profiles:create:binary'
  - 'profiles:create:json'
  - 'profiles:read'
  - 'profiles:id:read:json'
  - 'profiles:id:update:binaire'
  - 'profiles:id:read:binary'
  - 'profiles:id:update:json'
  - 'rules:read'
  - 'rules:create'
  - 'rules:id:read'
  - 'rulesfile:check'
  - 'rulesreport:id:read'
  - 'rulesreferential:id:read'
  - 'securityprofiles:create:json'
  - 'securityprofiles:read'
  - 'securityprofiles:id:read'
  - 'securityprofiles:id:update'
  - 'traceability:id:read'
  - 'traceabilitychecks:create'
  - 'workflows:read'
  - 'ingests:create'
  - 'ingests:local:create'
  - 'ingests:id:archivetransfertreply:read'
  - 'ingests:id:manifests:read'
  - 'switchindex:create'
  - 'reindex:create'
  - 'evidenceaudit:check'
  - 'archiveunitprofiles:create:binary'
  - 'archiveunitprofiles:create:json'
  - 'archiveunitprofiles:read'
  - 'archiveunitprofiles:id:read:json'
  - 'archiveunitprofiles:id:update:json'
```

(suite sur la page suivante)

(suite de la page précédente)

```
  - 'ontologies:create:binary'
  - 'ontologies:create:json'
  - 'ontologies:read'
  - 'ontologies:id:read:json'
  - 'ontologies:id:read:binary'
  - 'ontologies:id:update:json'
  - 'reclassification:update'
  - 'rectificationaudit:check'
  - 'storageaccesslog:read:binary'
  - 'objects:read'
  - 'elimination:analysis'
  - 'elimination:action'
  - 'forcepause:check'
  - 'removeforcepause:check'
  - 'probativevalue:check'
  - 'probativevalue:create'
  - 'accessionregisterssymbolic:read'
  - 'griffins:create'
  - 'preservationScenarios:create'
  - 'griffins:read'
  - 'griffin:read'
  - 'preservationScenarios:read'
  - 'preservationScenario:read'
  - 'preservation:update'
  - 'batchreport:id:read'
  - 'preservationreport:id:read'
  - 'logbookoperations:create'
  - 'computeInheritedRules:action'
  - 'computeInheritedRules:delete'
  - 'managementcontracts:create:json'
  - 'managementcontracts:read'
  - 'managementcontracts:id:read'
  - 'managementcontracts:id:update'
```

8.2.11.3 Opérations

- Démarrage du service

En tant qu'utilisateur root : `systemctl start vitam-security-internal`

- Arrêt du service

En tant qu'utilisateur root : `systemctl stop vitam-security-internal`

- Sauvegarde du service

Ce service ne nécessite pas de sauvegarde particulière.

- Supervision du service

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port>/vitam-security-internal/v1/status`

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port admin>/admin/v1/status`

- Exports

N/A

- gestion de la capacité

N/A

- actions récurrentes
- cas des batches

N/A

8.2.12 Logbook

8.2.12.1 Présentation

8.2.12.2 Logbook Exploitation

8.2.12.2.1 Configuration du Logbook

logbook.conf : fichier Yaml de configuration du serveur *logbook*. Celle-ci possède une propriété :

- **alertEvents** : configuration des alertes de sécurité

une alerte est déclenchée soit sur l'analyse du couple {evType,outCome} soit sur celle du {outDetail}

1. Dans le cas du déclenchement sur l'analyse du couple {evType, outCome}

```
- evType: 'CHECK_HEADER.CHECK_CONTRACT_INGEST'
  outcome: 'KO'
```

2. Dans le cas du déclenchement sur l'analyse du {outComeDetail}

```
- outDetail: 'CHECK_HEADER.CHECK_CONTRACT_INGEST.KO'
```

3. La liste des détections de l'alerte

- non conformité de la base des règles de gestion au référentiel enregistré (CHECK_RULES)
- refus d'entrée d'un SIP pour des raisons d'inadéquation de contrats (CHECK_HEADER.CHECK_CONTRACT_INGEST)
- soumission d'un SIP avec une classification incompatible avec la plateforme (CHECK_CLASSIFICATION_LEVEL)
- valeur de durée dans les régle de gestion inférieure à la durée minimum (CHECK_RULES.MAX_DURATION_EXCEEDS)
- refus d'un accès avec les droits personae (STP_PERSONAL_CERTIFICATE_CHECK)
- absence de sécurisation des journaux sur 12h (TODO)

Les fichiers de configuration sont gérés par les procédures d'installation ou de mise à niveau de l'environnement *VITAM*. Se référer au *DIN*.

Les fichiers de configuration sont définis sous `/vitam/conf/logbook`.

8.2.12.2.2 Fichier `logbook.conf`

```
# Configuration MongoDB
mongoDbNodes:
{% for server in groups['hosts_mongos_data'] %}
- dbHost: {{ hostvars[server]['ip_service'] }}
  dbPort: {{ mongodb.mongos_port }}
{% endfor %}
dbName: logbook
dbAuthentication: {{ mongodb.mongo_authentication }}
dbUserName: {{ mongodb['mongo-data'].logbook.user }}
dbPassword: {{ mongodb['mongo-data'].logbook.password }}
jettyConfig: jetty-config.xml
p12LogbookPassword: {{ keystores.timestamping.secure_logbook }}
p12LogbookFile: keystore_secure-logbook.p12
workspaceUrl: {{ vitam.workspace | client_url }}
processingUrl: {{ vitam.processing | client_url }}

# ElasticSearch
clusterName: {{ vitam_struct.cluster_name }}
elasticsearchNodes:
{% for server in groups['hosts_elasticsearch_data'] %}
- hostName: {{ hostvars[server]['ip_service'] }}
  tcpPort: {{ elasticsearch.data.port_tcp }}
{% endfor %}

#Basic Authentication
adminBasicAuth:
- userName: {{ admin_basic_auth_user }}
  password: {{ admin_basic_auth_password }}

## Configuration for logbook coherence check
# list of operations that generate LFC
opWithLFC: [
  "PROCESS_SIP_UNITARY",
  "FILINGSCHEME",
  "HOLDINGSCHEME",
  "UPDATE_RULES_ARCHIVE_UNITS",
  "PROCESS_AUDIT",
  "STP_UPDATE_UNIT"]
# list of events not declared in wf
opEventsNotInWf: [
  "STP_SANITY_CHECK_SIP",
  "SANITY_CHECK_SIP",
  "CHECK_CONTAINER",
  "STP_UPLOAD_SIP"
]
# list of events to skip for OP-LFC check
opLfcEventsToSkip: [
  "STP_SANITY_CHECK_SIP", "SANITY_CHECK_SIP", "CHECK_CONTAINER", "STP_UPLOAD_SIP",
↪"ATR_NOTIFICATION", "ROLL_BACK",
  "STORAGE_AVAILABILITY_CHECK", "ACCESSION_REGISTRATION",
  "ROLL_BACK", "ATR_NOTIFICATION", "COMMIT_LIFE_CYCLE_OBJECT_GROUP", "COMMIT_LIFE_
↪CYCLE_UNIT",
  "LIST_OBJECTGROUP_ID", "REPORT_AUDIT",
  "LIST_ARCHIVE_UNITS", "LIST_RUNNING_INGESTS"]

# Configuration des alertes de securite
alertEvents:
```

(suite sur la page suivante)

(suite de la page précédente)

```
- evType: 'CHECK_HEADER.CHECK_CONTRACT_INGEST'
  outcome: 'KO'
- evType: 'CHECK_RULES.MAX_DURATION_EXCEEDS'
  outcome: 'KO'
- evType: 'CHECK_RULES'
  outcome: 'KO'
- outDetail: 'CHECK_CLASSIFICATION_LEVEL.KO'
- outDetail: 'STP_PERSONAL_CERTIFICATE_CHECK.KO'

# Traceability params
operationTraceabilityTemporizationDelay: {{ vitam.logbook.
↪operationTraceabilityTemporizationDelay }}
lifecycleTraceabilityTemporizationDelay: {{ vitam.logbook.
↪lifecycleTraceabilityTemporizationDelay }}
lifecycleTraceabilityMaxEntries: {{ vitam.logbook.lifecycleTraceabilityMaxEntries }}
```

8.2.12.2.3 Fichier `functional-administration-client.conf`

```
serverHost: {{ vitam.functional_administration.host }}
serverPort: {{ vitam.functional_administration.port_service }}
```

8.2.12.2.4 Fichier `logbook-client.conf`

```
serverHost: {{ vitam.logbook.host }}
serverPort: {{ vitam.logbook.port_service }}
```

8.2.12.2.5 Fichier `securisationDaemon.conf`

```
tenants:  [ "{{ vitam_tenant_ids | join('", "') }}" ]
```

8.2.12.2.6 Fichier `storage-client.conf`

```
serverHost: {{ vitam.storageengine.host }}
serverPort: {{ vitam.storageengine.port_service }}
```

8.2.12.2.7 Fichier `traceabilityAudit.conf`

```
tenants:  [ "{{ vitam_tenant_ids | join('", "') }}" ]
nbDay: 1
timesEachDay: 24
```

8.2.12.3 Opérations

- Démarrage du service

En tant qu'utilisateur root : `systemctl start vitam-logbook`

- Arrêt du service

En tant qu'utilisateur root : `systemctl stop vitam-logbook`

- Sauvegarde du service

Ce service ne nécessite pas de sauvegarde particulière.

- Supervision du service

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port>/logbook/v1/status`

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port admin>/admin/v1/status`

- Exports

N/A

- gestion de la capacité

N/A

- actions récurrentes

- cas des batches

N/A

8.2.13 Metadata

8.2.13.1 Présentation

8.2.13.2 Configuration / fichiers utiles

8.2.13.2.1 Fichier `metadata.conf`

```
workspaceUrl: {{vitam.workspace | client_url}}
urlProcessing: {{vitam.processing | client_url}}

# Archive Unit Profile cache settings (max entries in cache & retention timeout in␣
↪seconds)
archiveUnitProfileCacheMaxEntries: {{ vitam.metadata.
↪archiveUnitProfileCacheMaxEntries }}
archiveUnitProfileCacheTimeoutInSeconds: {{ vitam.metadata.
↪archiveUnitProfileCacheTimeoutInSeconds }}

# Schema validator cache settings (max entries in cache & retention timeout in␣
↪seconds)
schemaValidatorCacheMaxEntries: {{ vitam.metadata.schemaValidatorCacheMaxEntries }}
schemaValidatorCacheTimeoutInSeconds: {{ vitam.metadata.
↪schemaValidatorCacheTimeoutInSeconds }}

# DIP purge service (in minutes)
dipTimeToLiveInMinutes: {{ vitam.metadata.dipTimeToLiveInMinutes }}

# TRANSFER purge service (in minutes)
transfersSIPTimeToLiveInMinutes: {{ vitam.metadata.transfersSIPTimeToLiveInMinutes }}
```

(suite sur la page suivante)

(suite de la page précédente)

```
# Configuration MongoDB
mongoDbNodes:
{% for server in groups['hosts_mongos_data'] %}
- dbHost: {{ hostvars[server]['ip_service'] }}
  dbPort: {{ mongodb.mongos_port }}
{% endfor %}
dbName: metadata
dbAuthentication: {{ mongodb.mongo_authentication }}
dbUserName: {{ mongodb['mongo-data'].metadata.user }}
dbPassword: {{ mongodb['mongo-data'].metadata.password }}

jettyConfig: jetty-config.xml

# ElasticSearch
clusterName: {{ vitam_struct.cluster_name }}
elasticsearchNodes:
{% for server in groups['hosts_elasticsearch_data'] %}
- hostName: {{ hostvars[server]['ip_service'] }}
  tcpPort: {{ elasticsearch.data.port_tcp }}
{% endfor %}

#Basic Authentication
adminBasicAuth:
- userName: {{ admin_basic_auth_user }}
  password: {{ admin_basic_auth_password }}
```

8.2.13.2.1.1 Paramétrage des caches

Metadata maintient en mémoire un ensemble de caches pour la gestion des données peu modifiées et qui interviennent lors des modifications de métadonnées (référentiels d'ontologie, schéma de donnée).

Cache du référentiel de l'ontologie :

- `ontologyCacheMaxEntries` : Nombre maximum d'objets à maintenir dans le cache (par défaut 100). Ce paramètre dépend du nombre de traitements actifs.
- `ontologyCacheTimeoutInSeconds` : Durée en secondes de rétention des objets en cache (par défaut 300, soit 5 minutes)

Cache du référentiel des profiles d'unités archivistiques :

- `archiveUnitProfileCacheMaxEntries` : Nombre maximum d'objets à maintenir dans le cache (par défaut 100). Ce paramètre dépend du nombre de traitements actifs.
- `archiveUnitProfileCacheTimeoutInSeconds` : Durée en secondes de rétention des objets en cache (par défaut 300, soit 5 minutes)

Cache des validateurs de schémas chargés en mémoire :

- `schemaValidatorCacheMaxEntries` : Nombre maximum d'objets à maintenir dans le cache (par défaut 100). Ce paramètre dépend du nombre de traitements actifs.
- `schemaValidatorCacheTimeoutInSeconds` : Durée en secondes de rétention des objets en cache (par défaut 300, soit 5 minutes)

8.2.13.2.2 Fichier `functional-administration-client.conf`

```
serverHost: {{ vitam.functional_administration.host }}
serverPort: {{ vitam.functional_administration.port_service }}
```

8.2.13.2.3 Fichier `storage-client.conf`

```
serverHost: {{ vitam.storageengine.host }}
serverPort: {{ vitam.storageengine.port_service }}
```

8.2.13.3 Opérations

- Démarrage du service

En tant qu'utilisateur root : `systemctl start vitam-metadata`

- Arrêt du service

En tant qu'utilisateur root : `systemctl stop vitam-metadata`

- Sauvegarde du service

Ce service ne nécessite pas de sauvegarde particulière.

- Supervision du service

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port>/metadata/v1/status`

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port admin>/admin/v1/status`

- Exports

N/A

- gestion de la capacité

N/A

- actions récurrentes

- cas des batches

N/A

8.2.14 Processing

8.2.14.1 Introduction

8.2.14.1.1 But de cette documentation

Le but de cette documentation est d'expliquer la configuration et l'exploitation de ce module.

8.2.14.2 Processing

Nom de l'image docker : **processing**

Dans cette image est déployé le module processing

8.2.14.2.1 Configuration du worker

Dans **/vitam/conf** :

1. **processing.conf** : Fichier Yaml de configuration du server *processing*. Il possède une propriété :

- **jettyConfig** : emplacement du ficher de configuration XML *jetty* (exemple jetty-config.xml)
- **urlWorkspace** : URL d'accès au service distant *workspace* (exemple http://localhost:8088)
- **urlMetadata** : URL d'accès au service distant *metadata* (exemple http://localhost:8088)

2. **logbook-client.conf** : Fichier de configuration du client qui communique avec le **logbook**. Il contient les propriétés suivantes :

- **serverHost** : host distant du service logbook
- **serverPort** : port distant du service logbook

3. **server-identity.conf** : identification du serveur
4. **logback.xml** : configuration des logs

8.2.14.2.2 Supervision du service

Contrôler le retour HTTP 200 et identité du serveur (cf *server-identity.conf*) sur l'URL `<protocole web https ou https>://<host>:<port>/processing/v1/status`

8.2.14.3 Configuration / fichiers utiles

Les fichiers de configuration sont gérés par les procédures d'installation ou de mise à niveau de l'environnement *VITAM*. Se référer au *DIN*.

Les fichiers de configuration sont définis sous `/vitam/conf/processing`.

8.2.14.3.1 Fichier `processing.conf`

```
urlMetadata: {{ vitam_struct | client_url }}
urlWorkspace: {{ vitam.workspace | client_url }}
jettyConfig: jetty-config.xml
workflowRefreshPeriod: 1
processingCleanerPeriod: 1
maxDistributionInMemoryBufferSize: {{vitam.processing.
↪maxDistributionInMemoryBufferSize | default(100000) }}
maxDistributionOnDiskBufferSize: {{vitam.processing.maxDistributionOnDiskBufferSize |␣
↪default(100000000) }}
```

8.2.14.3.2 Fichier `version.conf`

```
binaryDataObjectVersions:
- BinaryMaster
- Dissemination
- Thumbnail
- TextContent
```

(suite sur la page suivante)

(suite de la page précédente)

```
physicalDataObjectVersions:
- PhysicalMaster
- Dissemination
```

8.2.14.3.3 Fichier `storage-client.conf`

```
serverHost: {{ vitam.storageengine.host }}
serverPort: {{ vitam.storageengine.port_service }}
```

8.2.14.3.4 Fichier `metadata-client.conf`

```
serverHost: {{ vitam.metadata.host }}
serverPort: {{ vitam.metadata.port_service }}
```

8.2.14.4 Opérations

- Démarrage du service

En tant qu'utilisateur root : `systemctl start vitam-processing`

- Arrêt du service

En tant qu'utilisateur root : `systemctl stop vitam-processing`

- Sauvegarde du service

Ce service ne nécessite pas de sauvegarde particulière.

- Supervision du service

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port>/processing/v1/status`

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port admin>/admin/v1/status`

- Exports

N/A

- gestion de la capacité

N/A

- actions récurrentes

- cas des batches

N/A

8.2.15 Storage

8.2.15.1 Introduction

8.2.15.1.1 But de cette documentation

Le but de cette documentation est d'expliquer la configuration et l'exploitation des modules :

- **storage-engine**
- **storage-offer-default**

8.2.15.2 storage-engine

8.2.15.2.1 Présentation

Rôle :

- Stockage des données (Méta Données, Objets Numériques et journaux SAE et de l'archive)

Fonctions :

- Utilisation de stratégie de stockage (abstraction par rapport aux offres de stockage sous-jacentes)
- Gestion des différentes offres de stockage

8.2.15.2.2 Storage Engine

Nom de l'image docker : **storage-engine**

Dans cette image sont déployés :

- le moteur de stockage (storage-engine)
- l'implémentation du driver correspondant à l'offre de stockage par défaut (storage-offer-default)

8.2.15.2.2.1 Configuration du moteur de stockage

Dans **/vitam/conf** :

1. **storage-engine.conf** : Fichier Yaml de configuration du server *storage-engine*. Il possède une propriété :

- **urlWorkspace** : URL d'accès au service distant *workspace* (exemple http://localhost:8088)

2. **driver-location.conf** : Fichier Yaml de configuration du DriverManager, Il permet de définir l'emplacement où sont stockés les fichiers JAR contenant les implémentations des différents drivers pour les différentes offres. Il possède une seule propriété :

- **driverLocation** : emplacement des jars (chemin absolu de préférence)

3. **driver-mapping.conf** : Fichier Yaml de configuration du DriverMapper (persistance de l'association driver / offre). Pour le moment, ce fichier de configuration contient le chemin d'accès aux fichiers qui définissent le mapping driver<->offre, plus tard il évoluera sans doute pour prendre en compte des données en base et donc contenir la configuration d'accès à la base. Il contient deux propriétés :

- **driverMappingPath** : Définit l'emplacement des fichiers de persistance (au jourd'hui on a 1 seul driver/offre, donc 1 seul fichier de persistence sera présent). La propriété doit finir par « / ».
- **delimiter** : Définit le « délimiteur » (CSV style) des fichiers.

4. **static-offer.json** : Contient la description de l'offre "default" au format JSON (un jour sera sans doute dans une base de données). En PJ un exemple de ce fichier. La propriété baseUrl et parameters nécessitent d'être templaté. Et la propriété parameters doit contenir keystore, trustore et leur mot de passe que le storage driver va utiliser pour la vérification de l'authentication. Il s'agit de l'URL d'accès à l'offre de stockage "default". Exemple :

```
{
  "id" : "default",
  "baseUrl" : "https://localhost:8088",
  "parameters" : {
    "user" : "bob"
    "keyStore-keyPath": "src/test/resources/storage-test/tls/client/client.p12",
    "keyStore-keyPassword": "vitam2016",
    "trustStore-keyPath": "src/test/resources/storage-test/tls/server/truststore.jks",
    "trustStore-keyPassword": "tazerty",
    "referent": "true"
  }
}
```

To remove TLS support :

- change « https » to « http » in **baseUrl**

```
{
  "id" : "default",
  "baseUrl" : "http://localhost:8088",
  "parameters" : {
    "user" : "bob"
  }
}
```

To define « referent » offer :

- choose **exactly one** offer by adding parameter **referent**

```
[
  {
    "id" : "default",
    "baseUrl" : "http://localhost:8088",
    "parameters" : {
      "user" : "bob",
      "referent": "true"
    }
  },
  {
    "id" : "offer2",
    "baseUrl" : "http://localhost:8089",
    "parameters" : {
      "user" : "bob"
    }
  }
]
```

- change storage-default-offer.json to disable authentication

```
jettyConfig: jetty-config-nossl.xml
authentication : false
```

- change the jetty-config-nossl.xml of the offer (CAS Manager) to not include any TLS configuration

5. **static-strategy.json** : Contient les informations de la stratégie de stockage (1 seule pour le moment). Ce fichier n'est pas à modifier.

```
{
 "id" : "default",
 "hot" : {
   "copy" : 1,
   "offers" : [
     {"id" : "default"}
   ]
 }
}
```

6. **server-identity.conf** : identification du serveur
7. **logback.xml** : configuration des logs

8.2.15.2.2.2 Configuration du driver de l'offre de stockage par défaut

Dans **/vitam/data** :

1. **fr.gouv.vitam.storage.offers.workspace.driver.DriverImpl** : Il s'agit du fichier de persistence. Il contient l'identifiant de l'offre associée au driver (plus tard potentiellement DES offres associées) : *« default »*. Il DOIT être placé dans le répertoire défini dans le fichier *driver-mapping.conf*.

Dans **/vitam/lib** :

1. **storage-driver-default.jar** : Il s'agit d'un jar contenant l'implémentation du Driver vitam pour l'offre *« storage-offer-default »*. Ce jar DOIT être placé dans le dossier défini dans la propriété *driverLocation* du fichier *driver-location.conf*. Par défaut il est chargé en tant que dépendance du projet.

8.2.15.2.2.3 Supervision du service

Contrôler le retour HTTP 200 et identité du serveur (cf *server-identity.conf*) sur l'URL `<protocole web https ou https>://<host>:<port>/storage/v1/status`

8.2.15.2.3 Configuration / fichiers utiles

Les fichiers de configuration sont gérés par les procédures d'installation ou de mise à niveau de l'environnement *VITAM*. Se référer au *DIN*.

Les fichiers de configuration sont définis sous `/vitam/conf/storage-engine`.

8.2.15.2.3.1 Fichier `driver-location.conf`

```
driverLocation: {{ vitam_folder_lib }}
```

8.2.15.2.3.2 Fichier `driver-mapping.conf`

```
driverMappingPath: {{ vitam_folder_data }}/
delimiter: ;
```

8.2.15.2.3.3 Fichier `static-offer.json`

```
{% if vitam.storageofferdefault.https_enabled==true %}
    {% set protocol = 'https' %}
{% else %}
    {% set protocol = 'http' %}
{% endif %}
[
{% for item in all_used_offers %}
{
{% if item.id is defined %}
    "id" : "{{ item.id }}",
{% else %}
    "id" : "{{ item.name }}.service.{{ item.vitam_site_name |default(vitam_site_name)
↪}}.{{ consul_domain }}",
{% endif %}
    "baseUrl" : "{{ protocol }}://{{ item.name }}.service.{{ item.vitam_site_name
↪|default(vitam_site_name) }}.{{ consul_domain }}:{{ vitam.storageofferdefault.port_
↪service }}",
    {% if item.asyncRead is defined %} "asyncRead": {{item.asyncRead|lower }}, {%
↪endif %}
    "parameters" : {
        {% if vitam.storageofferdefault.https_enabled==true %}
        "keyStore-keyPath": "{{ vitam_folder_conf }}/keystore_storage.p12",
        "keyStore-keyPassword": "{{ keystores.client_storage.storage }}",
        "trustStore-keyPath": "{{ vitam_folder_conf }}/truststore_storage.jks",
        "trustStore-keyPassword": "{{ truststores.client_storage }}"
        {% endif %}
    }
}
{% if not loop.last %},
{% endif %}
{% endfor %}
]
```

8.2.15.2.3.4 Fichier `static-strategy.json`

```
[
    {
        "id" : "default",
        "offers" : [
{% for item in vitam_strategy %}
{% if item.id is defined %}
            {"id" : "{{ item.id }}"{% if item.referent is defined %}{% if item.
↪referent|lower == "true" %}, "referent" : true{% endif %}{% endif %}{% if item.
↪status is defined %}, "status" :  "{{ item.status| upper }}" {% endif %}}{% if not
↪loop.last %},{% endif %}
{% else %}
            {"id" : "{{ item.name }}.service.{{ item.vitam_site_name |default(vitam_
↪site_name) }}.{{ consul_domain }}"{% if item.referent is defined %}{% if item.
↪referent|lower == "true" %}, "referent" : true{% endif %}{% endif %}{% if item.
↪status is defined %}, "status" :  "{{ item.status| upper }}" {% endif %}}{% if not
↪loop.last %},{% endif %}
{% endif %}
{% endfor %}
```

(suite sur la page suivante)

(suite de la page précédente)

```
        ]
    }
    {% if other_strategies is defined %}
    {% for strategy_name, strategy_offers in other_strategies.iteritems() %}
    ,
    {
        "id" : "{{ strategy_name }}",
        "offers" : [
            {% for strategy_offer in strategy_offers %}
            {"id" : "{{ strategy_offer.name }}.service.{{ strategy_offer.vitam_site_
↪name |default(vitam_site_name) }}.{{ consul_domain }}"{% if strategy_offer.status␣
↪is defined %}, "status" :  "{{ strategy_offer.status| upper }}" {% endif %}}{% if␣
↪not loop.last %},{% endif %}
            {% endfor %}
        ]
    }
    {% endfor %}
    {% endif %}

]
```

8.2.15.2.3.5 Fichier `storage-engine.conf`

```
urlWorkspace: {{ vitam.workspace | client_url }}
timeoutMsPerKB: {{ vitam.storageengine.timeoutMsPerKB }}
jettyConfig: jetty-config.xml
zippingDirecorty: {{ vitam_folder_data }}/storage_archives
loggingDirectory: {{ vitam_folder_log }}
p12LogbookPassword: {{ keystores.timestamping.secure_storage }}
p12LogbookFile: keystore_{{ vitam_timestamp_usage }}.p12
storageTraceabilityOverlapDelay: {{ vitam.storageengine.
↪storageTraceabilityOverlapDelay }}
restoreBulkSize: {{ vitam.storageengine.restoreBulkSize }}
minBatchThreadPoolSize: {{ vitam.storageengine.minBatchThreadPoolSize }}
maxBatchThreadPoolSize: {{ vitam.storageengine.maxBatchThreadPoolSize }}
batchDigestComputationTimeout: {{ vitam.storageengine.batchDigestComputationTimeout }}
offerSynchronizationBulkSize: {{ vitam.storageengine.offerSynchronizationBulkSize }}
offerSyncThreadPoolSize: {{ vitam.storageengine.offerSyncThreadPoolSize }}
offerSyncNumberOfRetries: {{ vitam.storageengine.offerSyncNumberOfRetries }}
offerSyncFirstAttemptWaitingTime: {{ vitam.storageengine.
↪offerSyncFirstAttemptWaitingTime }}
offerSyncWaitingTime: {{ vitam.storageengine.offerSyncWaitingTime }}
#Basic Authentication
adminBasicAuth:
- userName: {{ admin_basic_auth_user }}
  password: {{ admin_basic_auth_password }}
```

8.2.15.2.4 Opérations

- Démarrage du service

En tant qu'utilisateur root : `systemctl start vitam-storage`

- Arrêt du service

En tant qu'utilisateur root : `systemctl stop vitam-storage`

- Sauvegarde du service

Ce service ne nécessite pas de sauvegarde particulière.

- Supervision du service

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port>/storage/v1/status`

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port admin>/admin/v1/status`

- Exports

N/A

- gestion de la capacité

N/A

- actions récurrentes

- cas des batches

N/A

8.2.15.2.4.1 access-log

Le log des accès est généré lors d'un accès à l'objet (fichier numérique), que ce soit par téléchargement de l'objet ou export d'un DIP. Les accès à l'unité archivistique ne sont pas concernés.

Exemple de log généré lors de l'export d'un DIP d'une unité archivistique ayant un GOT contenant un objet

```
{"eventDateTime":"2019-01-11T12:50:53.344","xRequestId":
↪"aeeaaaaaachfmo4dabyw6aliht3q74aaaaaq","applicationId":"MyApplicationId-ChangeIt",
↪"objectIdentifier":"aeaaaaaaaahk2vrsabz26alhywthyoaaaaba","size":"11","qualifier":
↪"BinaryMaster","version":"1","contextId":"CT-000001","contractId":"ContratTNR",
↪"archivesId":"aeaqaaaaaahk2vrsabz26alhywthzbaaaaea"}
```

Structure des logs :

- « eventDateTime » : date et heure de l'accès au format AAAA-MM-JJTHH :MM :SS.[digits de millisecondes]
- « xRequestId » : identifiant de l'opération d'export du DIP
- « applicationId » : identifiant de l'application ayant demandé l'export du DIP
- « objectIdentifier » : identifiant de l'objet auquel on a accédé
- « size » : taille en octets de l'objet
- « qualifier » : usage de l'objet
- « version » : version de l'usage de l'objet
- « contextId » : identifiant du contexte utilisé pour l'accès
- « contractId » : identifiant du contrat utilisé pour l'accès
- « archivesId » : identifiant de l'unité archivistique dont dépend le groupe d'objets contenant l'objet auquel on a accédé

Selon le paramétrage du contrat d'accès (AccessLog ACTIVE/INACTIVE), l'accès à un objet sera journalisé ou non. Par défaut, l'accès n'est pas journalisé.

Pour l'heure système en cours, ces fichiers sont présents sur les machines hébergeant le composant **storage** sous l'arborescence `/vitam/log/storage/access-log/`. Chaque fichier est nommé tel que :

`<tenant>_<date>_<id opération>.log`

Le *timer* systemD `vitam-storage-accesslog-backup` effectue la pérénisation sur offre de ces fichiers chaque heure. Dès lors, les *accesslog* sont accessibles dans des *containers* nommés `<environnement>_<tenant>_storageaccesslog`.

Exemple en stockage filesystem pour un environnement nommé int : `/vitam/data/offer/container/int_<tenant>_storageaccesslog/`

8.2.15.3 offer

8.2.15.3.1 Présentation

Ce composant est une déclinaison des offres de stockage sur FileSystem et CEPH.

Rôle :

- Fournir une offre de stockage par défaut permettant la persistance des objets sur un système de fichier local

Fonctions :

- Offre de stockage fournie par défaut
- Stockage simple des objets numériques sur un système de fichiers local

Par tenant VITAM déclaré, 17 *containers* sont créés :

- accessionregisterdetail
- accessionregistersymbolic
- backup
- backup_operation
- check_logbookreports
- dip
- distribution_reports
- logbook
- manifest
- object
- objectGroup
- report
- rules
- storageaccesslog
- storagelog
- storagetraceability
- unit

selon la norme `<vitam_site_name>_<tenant>_<container>` (R9 et plus) ou `<tenant>_<container>` (R7 et migrations depuis R7).

8.2.15.3.2 Storage Offer Default

Nom de l'image docker : **storage-offer-default**

Dans cette image est déployée l'offre de stockage par défaut utilisant le workspace.

8.2.15.3.2.1 Configuration de l'offre de stockage

1. **default-storage.conf** : Fichier Yaml de configuration du service. Contient les propriétés suivantes :

- **contextPath** : context path du server (mettre / par défaut)
- **storagePath** : chemin sur le filesystem sur lequel sont stockés les objects (/vitam/data).

2. **server-identity.conf** : identification du serveur
3. **logback.xml** : configuration des logs

8.2.15.3.2.2 Supervision du service

Contrôler le retour HTTP 200 et identité du serveur (cf *server-identity.conf*) sur l'URL `<protocole web https ou https>://<host>:<port>/offer/v1/status`

8.2.15.3.3 Configuration / fichiers utiles

Les fichiers de configuration sont gérés par les procédures d'installation ou de mise à niveau de l'environnement *VITAM*. Se référer au *DIN*.

Les fichiers de configuration sont définis sous `/vitam/conf/offer`.

8.2.15.3.3.1 Fichier `default-offer.conf`

```
contextPath: /
# Smile : TODO : remove storagePath from this file
storagePath: {{ vitam_folder_data }}
jettyConfig: jetty-config.xml
authentication : {{ vitam_struct.https_enabled }}
# Configuration MongoDB
mongoDbNodes:
{% for server in groups['hosts_mongos_offer'] %}
{% if hostvars[server]['mongo_cluster_name'] == offer_conf or inventory_hostname in␣
↪single_vm_hostnames %}
- dbHost: {{ hostvars[server]['ip_service'] }}
  dbPort: {{ mongodb.mongos_port }}
{% endif %}
{% endfor %}
dbName: offer
dbAuthentication: {{ mongodb.mongo_authentication }}
dbUserName: {{ mongodb[offer_conf].offer.user }}
dbPassword: {{ mongodb[offer_conf].offer.password }}
```

8.2.15.3.3.2 Fichier `default-storage.conf`

```
provider: {{ vitam_offers[offer_conf]["provider"] }}

{% if vitam_offers[offer_conf]["provider"] in ["filesystem","filesystem-hash"] %}
storagePath: {{ vitam_folder_data }}
{% endif %}
```

(suite sur la page suivante)

(suite de la page précédente)

```
{% if vitam_offers[offer_conf]["provider"] in ["openstack-swift","openstack-swift-v2",
↪"openstack-swift-v3"] %}
swiftKeystoneAuthUrl: {{ vitam_offers[offer_conf]["swiftKeystoneAuthUrl"] | default("
↪") }}
swiftDomain: {{ vitam_offers[offer_conf]["swiftDomain"] | default("") }}
swiftProjectName: {{ vitam_offers[offer_conf]["swiftProjectName"] | default("") }}
swiftUser: {{ vitam_offers[offer_conf]["swiftUser"] | default("") }}
swiftPassword: {{ vitam_offers[offer_conf]["swiftPassword"] | default("") }}
swiftUrl: {{ vitam_offers[offer_conf]["swiftUrl"] | default("") }}
swiftTrustStore: {{ vitam_folder_conf }}/truststore_{{ vitam_struct.vitam_component }}
↪.jks
swiftTrustStorePassword: {{ password_truststore }}
swiftMaxConnectionsPerRoute: {{ vitam_offers[offer_conf]["swiftMaxConnectionsPerRoute
↪"] | default(200) }}
swiftMaxConnections: {{ vitam_offers[offer_conf]["swiftMaxConnections"] |␣
↪default(1000) }}
swiftConnectionTimeout: {{ vitam_offers[offer_conf]["swiftConnectionTimeout"] |␣
↪default(200000) }}
swiftReadTimeout: {{ vitam_offers[offer_conf]["swiftReadTimeout"] | default(60000) }}
swiftHardRenewTokenDelayBeforeExpireTime: {{ vitam_offers[offer_conf][
↪"swiftHardRenewTokenDelayBeforeExpireTime"] | default(60) }}
swiftSoftRenewTokenDelayBeforeExpireTime: {{ vitam_offers[offer_conf][
↪"swiftSoftRenewTokenDelayBeforeExpireTime"] | default(300) }}
{% endif %}

{% if vitam_offers[offer_conf]["provider"] == "amazon-s3-v1" %}
s3RegionName: {{ vitam_offers[offer_conf]["s3RegionName"] }}
s3Endpoint: {{ vitam_offers[offer_conf]["s3Endpoint"] }}
s3AccessKey: {{ vitam_offers[offer_conf]["s3AccessKey"] }}
s3SecretKey: {{ vitam_offers[offer_conf]["s3SecretKey"] }}
s3PathStyleAccessEnabled: {{ vitam_offers[offer_conf]["s3PathStyleAccessEnabled"] |␣
↪default(true) }}
s3SignerType: {{ vitam_offers[offer_conf]["s3SignerType"] | default("") }}
s3MaxConnections: {{ vitam_offers[offer_conf]["s3MaxConnections"] | default(50) }}
s3ConnectionTimeout: {{ vitam_offers[offer_conf]["s3ConnectionTimeout"] |␣
↪default(10000) }}
s3SocketTimeout: {{ vitam_offers[offer_conf]["s3SocketTimeout"] | default(50000) }}
s3RequestTimeout: {{ vitam_offers[offer_conf]["s3RequestTimeout"] | default(0) }}
s3ClientExecutionTimeout: {{ vitam_offers[offer_conf]["s3ClientExecutionTimeout"] |␣
↪default(0) }}
s3TrustStore: {{ vitam_folder_conf }}/truststore_{{ vitam_struct.vitam_component }}.
↪jks
s3TrustStorePassword: {{ password_truststore }}
{% endif %}

{% if vitam_offers[offer_conf]["provider"] in ["tape-library"] %}
tapeLibraryConfiguration:
  inputFileStorageFolder: "{{ vitam_folder_data }}/offer/inputFiles"
  inputTarStorageFolder: "{{ vitam_folder_data }}/offer/inputTars"
  outputTarStorageFolder: "{{ vitam_folder_data }}/offer/outputTars"
  maxTarEntrySize: {{ vitam_offers[offer_conf]["tapeLibraryConfiguration"][
↪"maxTarEntrySize"] | default(100000) }}
  maxTarFileSize: {{ vitam_offers[offer_conf]["tapeLibraryConfiguration"][
↪"maxTarFileSize"] | default(1000000) }}
  useSudo: {{ vitam_offers[offer_conf]["tapeLibraryConfiguration"]["useSudo"] |␣
↪default('false') }}
```

(suite sur la page suivante)

(suite de la page précédente)

```
  forceOverrideNonEmptyCartridges: {{ vitam_offers[offer_conf][
↪"tapeLibraryConfiguration"]["forceOverrideNonEmptyCartridges"] | default('false') }}
  archiveRetentionCacheTimeoutInMinutes: {{ vitam_offers[offer_conf][
↪"tapeLibraryConfiguration"]["archiveRetentionCacheTimeoutInMinutes"] | default(30) }
↪}

  topology:
    buckets:
 {% for bucket in vitam_offers[offer_conf]["topology"]["buckets"] %}
        {{ bucket.name }}:
          tenants: {{ bucket.tenants }}
          tarBufferingTimeoutInMinutes: {{ bucket.tarBufferingTimeoutInMinutes }}
 {% endfor %}

  tapeLibraries:
{% for library in vitam_offers[offer_conf]["tapeLibraries"] %}
    {{ library.name }}:
      robots:
{% for robot in library.robots %}
        -
          device: {{ robot.device }}
          mtxPath: "{{  robot.mtxPath }}"
          timeoutInMilliseconds: {{ robot.timeoutInMilliseconds }}
{% endfor %}
      drives:
{% for drive in library.drives %}
        -
          index: {{ drive.index }}
          device: {{ drive.device }}
          mtPath: "{{ drive.mtPath }}"
          ddPath: "{{ drive.ddPath }}"
          tarPath: "{{ drive.tarPath }}"
          timeoutInMilliseconds: {{ drive.timeoutInMilliseconds }}
{% endfor %}
{% endfor %}
{% endif %}
```

L'arborescence de stockage des fichiers dans l'offre est décrite dans le *DAT*.

8.2.15.3.4 Opérations

- Démarrage du service

En tant qu'utilisateur root : `systemctl start vitam-offer`

- Arrêt du service

En tant qu'utilisateur root : `systemctl stop vitam-offer`

- Sauvegarde du service

Ce service ne nécessite pas de sauvegarde particulière.

- Supervision du service

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port>/offer/v1/status`

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port admin>/admin/v1/status`

- Exports

N/A

- gestion de la capacité

N/A

- actions récurrentes

- cas des batches

N/A

8.2.16 Technical administration

8.2.16.1 Présentation

8.2.17 Worker

8.2.17.1 Introduction

Le but de cette documentation est d'expliquer la configuration et l'exploitation du module **worker**.

8.2.17.2 Configuration / fichiers utiles

Les fichiers de configuration sont gérés par les procédures d'installation ou de mise à niveau de l'environnement *VITAM*. Se référer au *DIN*.

Les fichiers de configuration sont définis sous `/vitam/conf/worker`.

8.2.17.2.1 Fichier `batch-report-client.conf`

```
serverHost: {{ vitam.batchreport.host }}
serverPort: {{ vitam.batchreport.port_service }}
secure: {{ vitam.batchreport.https_enabled }}
```

8.2.17.2.2 Fichier `format-identifiers.conf`

Ce fichier permet de définir l'URL d'accès à Siegfried.

```
siegfried-local:
  type: SIEGFRIED
  client: http
  host: localhost
  port: {{ siegfried.port }}
  rootPath: {{ vitam_folder_tmp }}/
  versionPath: {{ vitam_folder_data }}/version/folder
```

8.2.17.2.3 Fichier `functional-administration-client.conf.j2`

Ce fichier permet de définir l'accès à functional-administration.

```
serverHost: {{ vitam.functional_administration.host }}
serverPort: {{ vitam.functional_administration.port_service }}
```

8.2.17.2.4 Fichier `metadata-client.conf`

Ce fichier permet de définir l'accès au metadata.

```
serverHost: {{ vitam.metadata.host }}
serverPort: {{ vitam.metadata.port_service }}
```

8.2.17.2.5 Fichier `storage-client.conf`

Ce fichier permet de définir l'accès au storage.

```
serverHost: {{ vitam.storageengine.host }}
serverPort: {{ vitam.storageengine.port_service }}
```

8.2.17.2.6 Fichier `verify-timestamp.conf`

```
#Configuration  - verify timestamp
p12LogbookPassword: {{ keystores.timestamping.secure_logbook }}
p12LogbookFile: keystore_secure-logbook.p12
```

8.2.17.2.7 Fichier `version.conf`

```
binaryDataObjectVersions:
- BinaryMaster
- Dissemination
- Thumbnail
- TextContent
physicalDataObjectVersions:
- PhysicalMaster
- Dissemination
```

8.2.17.2.8 Fichier `worker.conf`

Ce fichier permet de définir le paramétrage du composant worker.

```
# Configuration processing
# HERE MUST BE MY (WORKER) current configuration
registerServerHost: {{ ip_service }}
registerServerPort: {{ vitam_struct.port_service }}
# Configuration handler
```

(suite sur la page suivante)

(suite de la page précédente)

```
processingUrl: {{vitam.processing | client_url}}
urlMetadata: {{vitam.metadata | client_url}}
urlWorkspace: {{vitam.workspace | client_url}}
# Configuration jetty
jettyConfig: jetty-config.xml
#Configuration parallele
capacity: {{ vitam_worker_capacity }}
{% if vitam_worker_workerFamily is defined %}
workerFamily: {{ vitam_worker_workerFamily }}
{% endif %}

indexInheritedRulesWithAPIV2OutputByTenant: [ "{{  vitam.worker.api_output_index_
↪tenants | join('", "') }}" ]
indexInheritedRulesWithRulesIdByTenant: [ "{{  vitam.worker.rules_index_tenants |␣
↪join('", "') }}" ]

# Archive Unit Profile cache settings (max entries in cache & retention timeout in␣
↪seconds)
archiveUnitProfileCacheMaxEntries: {{ vitam.metadata.
↪archiveUnitProfileCacheMaxEntries }}
archiveUnitProfileCacheTimeoutInSeconds: {{ vitam.worker.
↪archiveUnitProfileCacheTimeoutInSeconds }}

# Schema validator cache settings (max entries in cache & retention timeout in␣
↪seconds)
schemaValidatorCacheMaxEntries: {{ vitam.metadata.schemaValidatorCacheMaxEntries }}
schemaValidatorCacheTimeoutInSeconds: {{ vitam.worker.
↪schemaValidatorCacheTimeoutInSeconds }}
```

Paramètres obligatoires :

- **processingUrl** : URL de connexion au compsoant Vitam processing
- **urlMetadata** : URL de connexion au composant VITAM metadata
- **urlWorkspace** : URL de connexion au composant VITAM workspace
- **registerServerHost** : host ou le worker déployé
- **registerServerPort** : port ou le worker déployé
- **jettyConfig** : le fichier config jetty associé au service du worker

Paramètres optionnels :

- **workerFamily** : la famille dont le worker appartant en fonction de tache exécutée
- **capacity** : capacité du worker en mode parallèle de tache (par défaut à 1 dans l'ansiblerie, si non définie)

8.2.17.3 Opérations

- Démarrage du service

En tant qu'utilisateur root : `systemctl start vitam-workspace`

- Arrêt du service

En tant qu'utilisateur root : `systemctl stop vitam-workspace`

- Sauvegarde du service

Ce service ne nécessite pas de sauvegarde particulière.

- Supervision du service

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port>/workspace/v1/status`

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port admin>/admin/v1/status`

- Exports

N/A

- gestion de la capacité

N/A

- actions récurrentes

- cas des batches

N/A

8.2.18 Workspace

8.2.18.1 Présentation

Rôle :

- Fourniture d'un espace pour l'échange de fichiers (et faire un appel par pointeur lors des appels entre composants) entre les différents composants de Vitam
- Fourniture d'un espace pour le stockage temporaire de données (exports DIP disponibles en téléchargement)

Fonctions :

- Utilisation du moteur de stockage dans un mode minimal (Opérations CREATE, READ, DELETE sur 1 seule offre de stockage)

8.2.18.2 Configuration / fichiers utiles

Les fichiers de configuration sont gérés par les procédures d'installation ou de mise à niveau de l'environnement *VITAM*. Se référer au *DIN*.

Les fichiers de configuration sont définis sous `/vitam/conf/workspace`.

8.2.18.2.1 Fichier `workspace.conf`

```
storagePath: {{ vitam_folder_data }}
jettyConfig: jetty-config.xml
provider: filesystem
```

8.2.18.3 Opérations

- Démarrage du service

En tant qu'utilisateur root : `systemctl start vitam-workspace`

- Arrêt du service

En tant qu'utilisateur root : `systemctl stop vitam-workspace`

- Sauvegarde du service

Ce service ne nécessite pas de sauvegarde particulière.

- Supervision du service

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port>/workspace/v1/status`

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port admin>/admin/v1/status`

- Exports

N/A

- gestion de la capacité

N/A

- actions récurrentes

- cas des batches

N/A

CHAPITRE 9

Intégration d'une application externe dans Vitam

9.1 Prérequis

L'application externe devra être en mesure de requêter les composants *VITAM* `ingest-external` et `access-external` sur leurs ports de service respectifs par le protocole HTTPS.

Il faut donc prévoir une ouverture de flux réseau pour le protocole TCP (selon l'infrastructure en place) sur les ports de service des composants *VITAM* `ingest-external` et `access-external`.

La sécurisation des connexions HTTP avec les applications externes est déléguée aux composants *VITAM* `ingest-external` et `access-external` (ou bien éventuellement à un reverse-proxy, selon l'infrastructure de déploiement *VITAM* retenue).

La création d'un certificat TLS client pour l'application externe est requise afin de permettre l'habilitation et l'authentification applicative des requêtes émises depuis l'application externe vers les composants *VITAM* ingest-external et access-external.

9.2 Intégration de certificats clients de VITAM

Note : Cette section décrit l'ajout de certificats *SIA* ou personnels (*Personae*) dans le cadre des procédures d'exploitation de la solution logicielle *VITAM*. L'intégration de certificats clients de *VITAM* au déploiement est décrite dans le document d'installation (*DIN*).

9.2.1 Authentification applicative SIA

Le certificat client de l'application externe doit être ajouté dans la base de données du composant `security-internal`, afin de permettre la gestion des habilitations et l'authentification applicative de l'application externe.

L'exemple suivant permet d'ajouter un certificat présent sous `/path/to/certificate` et associé au contexte applicatif portant l'identifiant `contexte_identifier`.

9.2.1.1 Ajout d'un certificat pour l'authentification applicative SIA

```
curl -XPOST -H "Content-Type:application/json" 'http://<ip admin security-internal>:
↪<port admin security-internal>/v1/api/identity' -d "
{
    \"contextId\":\"contexte_identifier\",
    \"certificate\":\"$(base64 /path/to/certificate)\"
}"
```

Note : Se repporter à la documentation sur la gestion des habilitations pour ce qui concerne l'ajout de contextes applicatifs.

Avertissement : Lorsque l'authentification du client, par le protocole TLS, est réalisée, la *CA* du certificat client doit être déployée dans le *truststore* des composants *VITAM* `ingest-external` et `access-external`. Dans ce cas de figure, suivre la procédure de la section *Mise à jour des certificats* (page 10), en ayant pris soin au préalable de déposer les certificats de la chaîne de certification client dans le répertoire *environments/certs/client-external/ca*. Si le certificat client est passé dans le *header* http (cas de l'authentification applicative par *header* http X-SSL-CLIENT-CERT), le certificat client n'est alors pas utilisé dans la négociation TLS et il n'est donc pas nécéssaire d'inclure la *CA* associée dans le *truststore* des composants *VITAM* `ingest-external` et `access-external`.

9.2.2 Authentification *Personae*

Le certificat personnel (*Personae*) doit être ajouté dans la base de données du composant `security-internal`, afin de permettre l'authentification renforcée de l'utilisateur.

Les exemples suivants permettent d'ajouter ou supprimer un certificat présent sous `/path/to/certificate`.

9.2.2.1 Ajout d'un certificat pour l'authentification *Personae*

```
curl -XPOST -H "Content-type: application/octet-stream" --data-binary @/path/to/
↪certificate 'http://<ip admin security-internal>:<port admin security-internal>/v1/
↪api/personalCertificate'
```

9.2.2.2 Suppression d'un certificat pour l'authentification *Personae*

```
curl -XDELETE -H "Content-type: application/octet-stream" --data-binary @/path/to/
↪certificate 'http://<ip admin security-internal>:<port admin security-internal>/v1/
↪api/personalCertificate'
```

9.3 Révocation de certificats clients de VITAM

La release « R8 » introduit une nouvelle fonctionnalité permettant la révocation des certificats *SIA* et *Personae* afin d'empecher des accès non autorisés aux *API* de la solution logicielle *VITAM* (vérification dans la couche https des *CRL*).

Le fonctionnement de la validation des certifcats de la solution logicielle *VITAM* *SIA* et *Personae* par *CRL* est le suivant :

- L'administrateur transmet à la solution logicielle *VITAM* le *CRL* d'un *CA* qui a émis le certificat présent dans la solution logicielle *VITAM*, via le point d'API suivant

```
http://{{ hosts_security_internal }}:{{vitam.security_internal.port_admin}}/v1/
↪api/crl
```

Prudence : La CRL fournie doit être obligatoirement au format DER (cf. http://www.ietf.org/rfc/rfc3280.txt »>RFC 3280 : *Internet X.509 Public Key Infrastructure Certificate and CRL Profile*)

Exemple :

```
curl -v -X POST -u {{ admin_basic_auth_user }}:{{ admin_basic_auth_password }}  http:/
↪/{{ hosts_security_internal }}:{{vitam.security_internal.port_admin}}/v1/api/crl -H
↪'Content-Type: application/octet-stream' --data-binary @/path/to/crl/my.crl
```

Le paramètre `adminUser` correspond à la valeur `admin_basic_auth_user` déclarée dans le fichier `vitam_security.yml`

Le paramètre `adminPassword` correspond à la valeur `admin_basic_auth_password` déclarée dans le fichier `vault-vitam.yml`

- Le système va contrôler tous les certificats (collections `identity.Certificate` et `identity.PersonalCertificate`) émis par le *IssuerDN* correspondant à la *CRL*, en vérifiant si ces derniers sont révoqués ou non. Si c'est le cas, alors la solution logicielle *VITAM* positionne le statut du certificat révoqué à **REVOKED**. Cela a pour conséquence le rejet de tout accès aux *API* *VITAM* avec utilisation du certificat révoqué (les filtres de sécurité émettront des exceptions dans les journaux de *log*).
- Une alerte de sécurité est émise dans les journaux en cas de révocation.

9.4 Déploiement des keystores

9.4.1 Vitam n'est pas encore déployé

Déployer Vitam en suivant la procédure indiquée dans le *DIN*.

9.4.2 Vitam est déjà déployé

Suivre la procédure de la section *Mise à jour des certificats* (page 10).

CHAPITRE 10

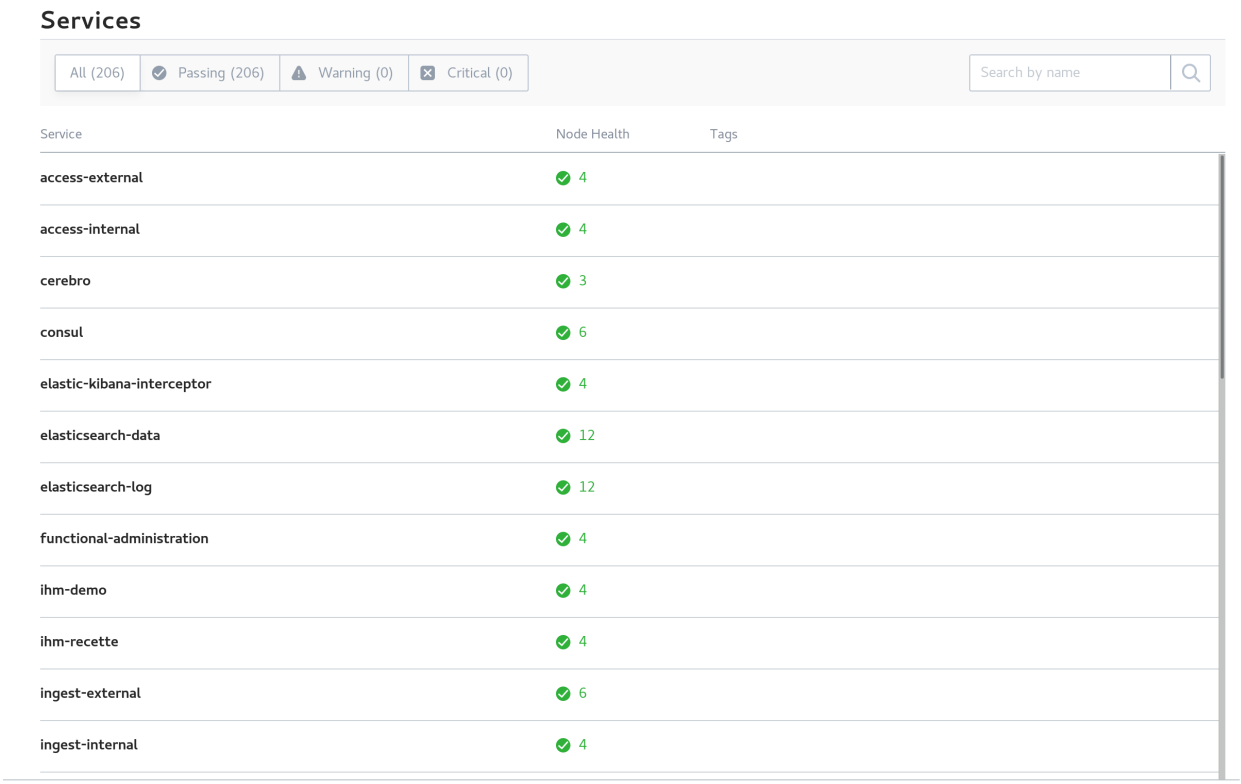
Aide à l'exploitation

10.1 Analyse de premier niveau

Cette section a pour but de présenter les premiers outils à utiliser pour réaliser une analyse de premier niveau, en cas de problème avec la solution logicielle *VITAM*.

10.1.1 Etat par Consul

Se connecter à l'IHM de Consul et recenser les états des composants de la solution logicielle *VITAM*.



A l'heure actuelle, tous les composants doivent avoir un statut de couleur verte. Si ce n'est pas le cas :

1. seul un composant est KO, alors redémarrer le composant incriminé
2. si plusieurs services sont KO, suivre la procédure de redémarrage de VITAM
3. si tous les « check-DNS » (visible dans le détail des checks de chaque service) sont KO, s'assurer que, sur les machines hébergeant VITAM, le fichier `/etc/resolv.conf` contient, en début de fichier, la ligne : `nameserver 127.0.0.1`.

10.1.2 Etat par Kibana

Se connecter à Kibana, aller dans « Dashboards ». Cliquer sur le bouton « Load Saved Dashboard » et sélectionner « Composants VITAM ». Eventuellement, changer la résolution (en haut à droite, par défaut, réglé sur les 15 dernières minutes).

Sur « pie-logback-error-level », cliquer sur la section de camembert d'intérêt (ERROR) et regarder, en bas de page, les éventuelles erreurs remontées dans Kibana.

10.2 Playbook ansible pour échanger avec le support

Afin de simplifier et minimiser les échanges, un playbook à fins d'exploitation/support a été développé.

Celui-ci comprend :

- la récupération des informations machines de la solution logicielle *VITAM*
- l'état Consul des composants
- la récupération des traces applicatives (fichiers log) de moins de 2 jours
- l'état des clusters Elasticsearch

- la possibilité, au choix de l'exploitant, de fournir également les clés publiques des certificats

et compacte l'ensemble des données collectées, tout en purgeant le répertoire temporaire de récupération des données. Ce fichier est alors à envoyer par *mail* au support.

La commande pour générer le fichier est à lancer depuis le répertoire `deployment` :

```
ansible-playbook -i environments/<fichier d'inventaire> ansible-vitam-exploitation/
troubleshoot.yml --vault-password-file vault_pass.txt
```

10.3 Identification des AU non conformes

Le schéma JSON des AU a été corrigé et rationalisé. Toute donnée issue d'un Ingest d'une release précédente est conforme à cette expression corrigée du schéma. Dans les versions précédentes, le DSL autorisait des modifications non conformes au SEDA. Ce n'est plus possible dans la présente version.

Si des modifications non conformes ont eu lieu, alors des AU non conformes peuvent donc se retrouver en base. Lorsque cela arrive : * L'export DIP de l'AU peut échouer. * La modification d'un champ A rapportera une erreur sur un champs B non-conforme. La correction de ce problème se fera avec une requête DSL qui modifie le champ A et le champs B (en lui donnant une valeur conforme). * Maintenant les champs SEDA sont soit des tableaux, soit des valeurs simples (string ou objet). Certains champs qui pouvaient s'écrire sous forme de valeurs non tabulaires, doivent maintenant être écrits sous forme de tableau, même s'ils n'ont qu'un seul élément. Ex : Coverage.Spatial.

Cette procédure scriptée permet de détecter l'ensemble des AU non conformes. Les retours associés pourront être transmis au support VITAM (assistance@programmevitam.fr).

La commande pour générer le fichier est à lancer depuis le répertoire `deployment` :

```
ansible-playbook -i environments/<fichier d'inventaire> ansible-vitam-exploitation/
check_unit_compatibility.yml --vault-password-file vault_pass.txt
```

A l'issue, si des *AU* sont considérées en erreur, se rapprocher du métier pour réaliser une première analyse / tentative de correction, par correction des données incorrectes.

Si l'erreur persiste, contacter le support.

CHAPITRE 11

Questions Fréquemment Posées

11.1 Présentation

Cette section a vocation à répertorier les différents problèmes rencontrés et apporter la solution la plus appropriée ; elle est amenée à être régulièrement mise à jour pour répertorier les problèmes rencontrés.

11.2 Retour d'expérience / cas rencontrés

11.2.1 Crash rsyslog, code killed, signal : BUS

Il a été remarqué chez un partenaire du projet Vitam, que rsyslog se faisait *killer* peu après son démarrage par le signal SIGBUS. Il s'agit très probablement d'un bug rsyslog <= 8.24 https://github.com/rsyslog/rsyslog/issues/1404

Pour fixer ce problème, il est possible d'upgrader rsyslog sur une version plus à jour en suivant cette documentation :

- Centos [18]
- Debian [19]

11.2.2 Mongo-express ne se connecte pas à la base de données associée

Si mongoDB a été redémarré, il faut également redémarrer mongo-express.

11.2.3 Elasticsearch possède des shard non alloués (état « UNASSIGNED »)

Lors de la perte d'un noeud d'un cluster elasticseach, puis du retour de ce noeud, certains shards d'elasticseach peuvent rester dans l'état `UNASSIGNED` ; dans ce cas, cerebro affiche les shards correspondant en gris (au-dessus des noeuds) dans la vue « cluster », et l'état du cluster passe en « yellow ». Il est possible d'avoir plus d'informations sur la

[18] https://www.rsyslog.com/rhelcentos-rpms/
[19] https://www.rsyslog.com/debian-repository/

cause du problème via une requête `POST` sur l'API elasticsearch `_cluster/reroute?explain`. Si la cause de l'échec de l'assignation automatique a été résolue, il est possible de relancer les assignations automatiques en échec via une requête `POST` sur l'API `_cluster/reroute?retry_failed`. Dans le cas où l'assignation automatique ne fonctionne pas, il est nécessaire de faire l'assignation à la main pour chaque shard incriminé (requête `POST` sur `_cluster/reroute`) :

```
{
        "commands": [
                {
                        "allocate": {
                                "index": "topbeat-2016.11.22",
                                "shard": 3,
                                "node": "vitam-iaas-dblog-01.int"
                        }
                }
        ]
}
```

Cependant, un shard primaire ne peut être réalloué de cette manière (il y a risque de perte de données). Si le défaut d'allocation provient effectivement de la perte puis de la récupération d'un noeud, et que TOUS les noeuds du cluster sont de nouveaux opérationnels et dans le cluster, alors il est possible de forcer la réallocation sans perte.

```
{
        "commands": [
                {
                        "allocate": {
                                "index": "topbeat-2016.11.22",
                                "shard": 3,
                                "node": "vitam-iaas-dblog-01.int",
                                "allow_primary": "true"
                        }
                }
        ]
}
```

Sur tous ces sujets, Cf. la documentation officielle [20].

11.2.4 Elasticsearch possède des shards non initialisés (état « INITIALIZING »)

Tout d'abord, il peut être difficile d'identifier les shards en questions dans cerebro ; une requête HTTP `GET` sur l'API `_cat/shards` permet d'avoir une liste plus compréhensible. Un shard non initialisé correspond à un shard en cours de démarrage (Cf. une ancienne page de documentation [21]. Si les shards non initialisés sont présents sur un seul noeud, il peut être utile de redémarrer le noeud en cause. Sinon, une investigation plus poussée doit être menée.

11.2.5 MongoDB semble lent

Pour analyser la performance d'un cluster MongoDB, ce dernier fournit quelques outils permettant de faire une première analyse du comportement : mongostat [22] et mongotop [23] .

Dans le cas de VITAM, le cluster MongoDB comporte plusieurs shards. Dans ce cas, l'usage de ces deux commandes peut se faire :

https://www.elastic.co/guide/en/elasticsearch/reference/current/cluster-reroute.html
https://www.elastic.co/guide/en/elasticsearch/reference/1.4/states.html
https://docs.mongodb.com/manual/reference/program/mongostat/
https://docs.mongodb.com/manual/reference/program/mongotop/

- soit sur le cluster au global (en pointant sur les noeuds mongos) : cela permet d'analyser le comportement global du cluster au niveau de ses points d'entrées ;

```
mongostat --host <ip_service> --port 27017 --username vitamdb-admin --
↪password <password ; défaut : azerty> --authenticationDatabase admin
mongotop --host <ip_service> --port 27017 --username vitamdb-admin --
↪password <password ; défaut : azerty> --authenticationDatabase admin
```

- soit directement sur les noeuds de stockage (mongod) : cela donne des résultats plus fins, et permet notamment de séparer l'analyse sur les noeuds primaires & secondaires d'un même replicaset.

```
mongotop --host <ip_service> --port 27019 --username vitamdb-localadmin --
↪password <password ; défaut : qwerty> --authenticationDatabase admin
mongostat --host <ip_service> --port 27019 --username vitamdb-localadmin -
↪-password <password ; défaut : qwerty> --authenticationDatabase admin
```

D'autres outils sont disponibles directement dans le client mongo, notamment pour troubleshooter les problèmes dûs à la réplication [24] :

```
mongo --host <ip_service> --port 27019 --username vitamdb-localadmin --password
↪<password ; défaut : qwerty> --authenticationDatabase admin
> rs.printSlaveReplicationInfo()
> rs.printReplicationInfo()
> db.runCommand( { serverStatus: 1 } )
```

D'autres commandes plus complètes existent et permettent d'avoir plus d'informations, mais leur analyse est plus complexe :

```
# returns a variety of storage statistics for a given collection
> use metadata
> db.stats()
> db.runCommand( { collStats: "Unit" } )
```

Enfin, un outil est disponible en standard afin de mesurer des performances des lecture/écritures avec des patterns proches de ceux utilisés par la base de données (mongoperf [25]) :

```
echo "{nThreads:16,fileSizeMB:10000,r:true,w:true}" | mongoperf
```

11.2.6 Les shards de MongoDB semblent mal équilibrés

Normalement, un processus interne à MongoDB (le `balancer`) s'occupe de déplacer les données entre les shards (par `chunk`) pour équilibrer la taille de ces derniers. Les commandes suivantes (à exécuter dans un shell mongo sur une instance mongos - attention, ces commandes ne fonctionnent pas directement sur les instances mongod) permettent de s'assurer du bon fonctionnement de ce processus :

- `sh.status()` : donne le status du sharding pour le cluster complet ; c'est un bon premier point d'entrée pour connaître l'état du balancer.
- `use <dbname>`, puis `db.<collection>.getShardDistribution()`, en indiquant le bon nom de base de données (ex : `metadata`) et de collection (ex : `Unit`) : donne les informations de répartition des chunks dans les différents shards pour cette collection.

https://docs.mongodb.com/manual/tutorial/troubleshoot-replica-sets
https://docs.mongodb.com/manual/reference/program/mongoperf/

11.2.7 L'importation initiale (profil de sécurité, certificats) retourne une erreur

Les playbooks d'initialisation importent des éléments d'administration du système (profils de sécurité, certificats) à travers des APIs de la solution VITAM. Cette importation peut être en échec, par exemple à l'étape `TASK [init_contexts_and_security_profiles : Import admin security profile to functionnal-admin]`, avec une erreur de type 400. Ce type d'erreur peut avoir plusieurs causes, et survient notamment lors de redéploiements après une première tentative non réussie de déploiement ; même si la cause de l'échec initial est résolue, le système peut se trouver dans un état instable. Dans ce cas, un déploiement complet sur environnement vide est nécessaire pour revenir à un état propre.

Une autre cause possible ici est une incohérence entre l'inventaire, qui décrit notamment les offres de stockage liées aux composants offer, et le paramétrage `vitam_strategy` porté par le fichier `offers_opts.yml`. Si une offre indiquée dans la stratégie n'existe nulle part dans l'inventaire, le déploiement sera en erreur. Dans ce cas, il faut remettre en cohérence ces paramètres et refaire un déploiement complet sur environnement vide.

11.2.8 Problème d'ingest et/ou d'access

Si vous repérez un message de ce type dans les log *VITAM* :

```
fr.gouv.vitam.common.security.filter.AuthorizationWrapper.
↪checkTimestamp(AuthorizationWrapper.java:117) : [vitam-env-int8-app-04.vitam-
↪env:storage:239079175] Timestamp check failed
```

Il faut vérifier / corriger l'heure des machines hébergeant la solution logicielle *VITAM* ; un *delta* de temps supérieur à 10s a été détecté entre les machines.

11.3 Erreur d'inconsistence des données MongoDB / ES

En cas de détection d'un problème de synhronisation des données entre les bases de données Elasticsearch-data (cluster d'indexation dédié aux données métier) et les bases de données MongoDB-data (replicaset MongoDB stockant les données métier de Vitam) avec un message d'erreur du type : « An internal data consistency error has been detected », la procédure suivante pourra être appliquée : *Réindexation* (page 39).

CHAPITRE 12

Annexes

12.1 Cycle de vie des certificats

Le tableau ci-dessous indique le mode de fonctionnement actuel pour les différents certificats et *CA*. Précisions :

- Les « procédures par défaut » liées au cycle de vie des certificats dans la présente version de la solution *VITAM* peuvent être résumées ainsi :
  - Création : génération par *PKI* partenaire + copie dans répertoires de déploiement + script `generate_stores.sh` + déploiement ansible
  - Suppression : suppression dans répertoires de déploiement + script `generate_stores.sh` + déploiement ansible
  - Renouvellement : regénération par *PKI* partenaire + suppression / remplacement dans répertoires de déploiement + script `generate_stores.sh` + redéploiement ansible
- Il n'y a pas de contrainte au niveau des *CA* utilisées (une *CA* unique pour tous les usages *VITAM* ou plusieurs *CA* séparées – cf. *DAT*). On appelle ici :
  - « *PKI* partenaire » : *PKI* / *CA* utilisées pour le déploiement et l'exploitation de la solution *VITAM* par le partenaire.
  - « *PKI* distante » : *PKI* / *CA* utilisées pour l'usage des frontaux en communication avec le back office *VITAM*.

Classe	Type	Usages	Origine	Création	Suppression	Renouvellement
Interne	*CA*	ingest & access	*PKI* partenaire	*proc. par défaut*	*proc. par défaut*	*proc. par défaut*
Interne	*CA*	offer	*PKI* partenaire	*proc. par défaut*	*proc. par défaut*	*proc. par défaut*
Interne	Certif	Horodatage	*PKI* partenaire	*proc. par défaut*	*proc. par défaut*	*proc. par défaut*
Interne	Certif	Storage (*Swift*)	Offre de stockage	*proc. par défaut*	*proc. par défaut*	*proc. par défaut*
Interne	Certif	Storage (s3)	Offre de stockage	*proc. par défaut*	*proc. par défaut*	*proc. par défaut*
Interne	Certif	ingest	*PKI* partenaire	*proc. par défaut*	*proc. par défaut*	*proc. par défaut*
Interne	Certif	access	*PKI* partenaire	*proc. par défaut*	*proc. par défaut*	*proc. par défaut*
Interne	Certif	offer	*PKI* partenaire	*proc. par défaut*	*proc. par défaut*	*proc. par défaut*
Interne	Certif	Timestamp	*PKI* partenaire	*proc. par défaut*	*proc. par défaut*	*proc. par défaut*
IHM demo	*CA*	ihm-demo	*PKI* partenaire	*proc. par défaut*	*proc. par défaut*	*proc. par défaut*
IHM demo	Certif	ihm-demo	*PKI* partenaire	*proc. par défaut*	*proc. par défaut*	*proc. par défaut*
SIA	*CA*	Appel *API*	*PKI* distante	*proc. par défaut* (*PKI* distante)	*proc. par défaut*	*proc. par défaut* (*PKI* distante)+recharger Certifs
SIA	Certif	Appel *API*	*PKI* distante	Génération + copie répertoire + deploy(par la suite appel *API* d'insertion)	Suppression Mongo	Suppression Mongo + *API* d'insertion
Personae	Certif	Appel *API*	*PKI* distante	*API* ajout	*API* suppression	*API* suppression + *API* ajout

Remarques :

- Lors d'un renouvellement de *CA SIA*, il faut s'assurer que les certificats qui y correspondaient soient retirés de MongoDB et que les nouveaux certificats soient ajoutés par le biais de l'" *API* dédiée.
- Lors de toute suppression ou remplacement de certificats *SIA*, s'assurer que la suppression ou remplacement des contextes associés soit également réalisé.
- L'expiration des certificats n'est pas automatiquement prise en charge par la solution *VITAM* (pas de notification en fin de vie, pas de renouvellement automatique). Pour la plupart des usages, un certificat expiré est proprement rejeté et la connexion ne se fera pas ; les seules exceptions sont les certificats *Personae*, pour lesquels la validation de l'arborescence *CA* et des dates est à charge du front office en interface avec *VITAM*.

12.2 Gestion des anomalies en production

Les anomalies empêchant le bon fonctionnement de la solution VITAM déjà déployée dans un système en production sont gérées par le programme VITAM selon un processus dédié. Il reprend la terminologie du « Contrat de Service VITAM ».

12.2.1 Numérotation des versions

A partir de la version 1 de la solution VITAM, la numérotation des versions du logiciel est du type *X.Y.Z(-P)* selon les principes suivants :

- X : version majeure de la solution VITAM. Elle suit le calendrier des versions majeures, construit de concert avec les partenaires.
- Y : version mineure de la solution VITAM. Elle suit le calendrier des itérations, typiquement une itération dure trois semaines.
- Z : version *bugfix* de la solution VITAM. Elle suit le calendrier des itérations, typiquement une itération à chaque trois semaines.
- Seules les versions maintenues continuent de bénéficier de nouvelles versions *bugfix*.
- P : patch de la solution VITAM. Un patch correspond à la mise à disposition, entre deux releases, de binaires et/ou fichiers de configuration et de déploiement, pour corriger des bugs bloquants.
- Seules les versions maintenues continuent de bénéficier de patchs.

12.2.2 Mise à disposition du logiciel

La solution VITAM est mise à disposition des partenaires selon le calendrier suivant :

- Des *releases* sont mises à disposition des partenaires et du grand public régulièrement, typiquement une release pour cinq itérations de développement. Il s'agit alors de la version mineure courante. Pour rappel, la version 1.0.0 correspond à la release 6 (*R6*).
- Les versions *bugfix* de chaque version maintenue sont mises à disposition des partenaires et du grand public régulièrement, à chaque itération (s'il y a eu des anomalies corrigées dans la période).
- Les patchs de chaque version maintenue sont mis à disposition des partenaires à chaque fois qu'une anomalie de production critique est identifiée et corrigée. Les correctifs correspondant aux patchs sont ensuite inclus dans une version *bugfix* ultérieure.

12.2.3 Gestion des patchs

L'objectif d'un patch est de rétablir au plus vite le fonctionnement en production des systèmes partenaires. La livraison se limite ainsi aux packages (RPM / DEB) concernés par la correction, avec les fichiers de déploiement et de configuration nécessaires. Les instructions pour « patcher » l'applicatif sont également mises à disposition, en fonction du périmètre impacté (simple arrêt / relance ; purges ; scripts de déploiement. . .).

Les patchs sont mis à disposition des partenaires sur un dépôt en ligne. L'objectif est d'offrir la possibilité pour les partenaires d'automatiser la récupération des packages mis à jour, et éventuellement de pouvoir reconstituer un packaging complet de Vitam.

Note : Ce choix de gestion de patchs implique des numéros de version qui pourront être différents entre chaque paquet. Le réalignement se fait au niveau des versions *bugfix* ou mineures.

La mise à disposition du code source du patch est considérée comme moins critique et se réalise dans un second temps, sur Github.

Table des figures

Liste des tableaux

Index