



VITAM - Documentation d'exploitation

Version 1.10.0-1

VITAM

nov. 19, 2018

Table des matières

1	Introduction	1
1.1	But de cette documentation	1
1.2	Destinataires de ce document	1
2	Rappels	2
2.1	Information concernant les licences	2
2.2	Documents de référence	2
2.2.1	Documents internes	2
2.2.2	Référentiels externes	2
2.3	Glossaire	2
3	Expertises requises	4
4	Architecture de la solution logicielle VITAM	5
5	Exploitation globale	7
5.1	Gestion des accès	7
5.2	Portails d'administration	7
5.2.1	Technique	7
5.2.2	Fonctionnel	7
5.3	Paramétrage & configuration	7
5.3.1	Mise à niveau de la configuration de l'environnement	8
5.3.1.1	Mise à jour du nombre de tenants	8
5.3.1.2	Mise à jour des paramètres JVM	8
5.4	Déploiement / mises à jour	8
5.4.1	Mise à jour des certificats	8
5.4.2	Mise à jour de la solution vitam	8
5.4.3	Ajouter un/des instances de services VITAM	9
5.5	Interruption / maintenance	9
5.5.1	Procédure d'arrêt	9
5.5.2	Procédure de démarrage	10
5.5.3	Procédure de statut	10
5.5.4	Procédure de maintenance / indisponibilité de VITAM	10
5.5.5	Procédure de maintenance liée aux <i>timers systemD</i>	10
5.5.6	Procédure de maintenance sur les composants d'administration	10
5.5.7	Procédure de maintenance des <i>IHM</i>	11
5.5.8	Procédure de maintenance des <i>Bases de données métier</i>	11

5.6	Sauvegarde / restauration	11
5.6.1	Sauvegarde	11
5.6.1.1	mongoDB	11
5.6.1.2	Elasticsearch	12
5.6.2	Restauration	12
5.6.2.1	mongoDB	12
5.6.2.2	Elasticsearch	13
5.7	Batches et traitements	13
5.7.1	Curator	13
5.7.2	Sécurisation des journaux d'opérations	13
5.7.3	Sécurisation des cycles de vie	13
5.7.4	Sécurisation des offres de stockages	13
5.8	Sauvegarde des données graphe (Log shipping)	14
5.8.1	Déclenchement	14
5.8.1.1	Reconstruction des données graphe :	14
5.9	Recalcul des données graphe	15
5.9.1	Déclenchement	15
5.10	Reconstruction	16
5.10.1	Déclenchement	16
5.10.1.1	Cas du site primaire :	16
5.10.1.2	Cas du site secondaire :	16
5.11	Plan de Reprise d'Activité (PRA)	17
5.11.1	Déclenchement	17
5.11.2	Retour en situation nominale	18
5.11.3	Déclenchement	18
5.12	Resynchronisation d'une offre	19
5.12.1	Décommissionnement d'une offre	19
5.12.2	Ajout d'une offre	20
5.12.3	Resynchronisation totale d'une offre	22
5.13	Procédure d'exploitation suite a la création ou la modification d'une ontologie	23
5.13.1	Création d'une ontologie	23
5.13.2	Changement de type d'une ontologie existante	23
5.14	Procédure d'exploitation pour la mise en pause forcée d'une opération	24
5.14.1	Mise en pause forcée	24
5.14.2	Sortie de la mise en pause forcée	25
5.15	Procédure d'exploitation pour la révocation des certificats SIA et Personae	26
5.15.1	Procédure de révocation	26
6	Suivi de l'état du système	27
6.1	Veille et patches sécurité	27
6.2	Métriques	27
6.2.1	Configuration	27
6.2.1.1	Registres	27
6.2.1.2	Reporters	28
6.2.1.3	Fichier de configuration	28
6.2.2	Métier	28
6.2.3	Métriques techniques	28
6.2.3.1	Métriques système critiques	28
6.2.3.2	Indicateurs de SLA	28
6.2.3.3	Indicateurs de performance	28
6.2.4	Visualisation	28
6.2.4.1	Discover	29
6.2.4.2	Visualize	29
6.2.4.3	Dashboards	30

6.3	API de de supervision	31
6.3.1	Détail	31
6.3.1.1	/admin/v1/status	31
6.3.1.2	/admin/v1/version	32
6.3.1.3	/admin/v1/autotest	33
6.4	Logs	34
6.4.1	Changement des règles de log	34
6.5	Audit	35
6.6	Gestion de la capacité	35
6.7	Suivi de l'état de sécurité	35
6.8	Alerting	35
6.8.1	Système	35
6.8.2	Applicatif	36
6.9	Suivi des Workflows	36
6.9.1	IHM	36
6.9.2	Appels REST	36
6.9.3	Worklow en FATAL	36
6.9.3.1	Plugins et Handlers	36
6.9.3.2	Distributeur	37
6.9.3.3	Processing - State Machine	38
6.9.4	Redémarrer un processus en cas de pause	38
6.9.4.1	Trouver la cause	38
6.9.4.2	Relancer le Workflow	38
6.9.4.2.1	Vérifier les inputs	38
6.9.4.2.2	Rejouer une étape	38
6.9.4.2.3	Prochaine étape	38
6.9.4.2.4	Finaliser le workflow	38
6.10	Cohérence des journaux	39
6.10.1	Lancement	39
6.10.2	Résultat	39
6.11	Liste des timers systemd	39
6.11.1	Timers de maintenance des index elasticsearch-log	39
6.11.1.1	vitam-curator-metrics-indexes	40
6.11.1.2	vitam-curator-close-old-indexes	40
6.11.1.3	vitam-curator-delete-old-indexes	40
6.11.2	Timers de gestion des journaux (preuve systématique)	40
6.11.2.1	vitam-storage-log-backup	41
6.11.2.2	vitam-storage-log-traceability	41
6.11.2.3	vitam-traceability-operations	41
6.11.2.4	vitam-traceability-lfc-unit	41
6.11.2.5	vitam-traceability-lfc-objectgroup	42
6.11.3	Timers d'audit interne VITAM	42
6.11.3.1	vitam-traceability-audit	42
6.11.3.2	vitam-rule-management-audit	42
6.11.4	Timer relatif aux liens symboliques de <i>accession register</i>	42
6.11.4.1	vitam-create-accession-register-symbolic	42
6.11.5	Timers de reconstruction VITAM	43
6.11.5.1	vitam-functional-administration-reconstruction	43
6.11.5.2	vitam-logbook-reconstruction	43
6.11.5.3	vitam-metadata-reconstruction	43
6.11.5.4	vitam-metadata-store-graph	44
7	Exploitation des composants de la solution logicielle VITAM	45
7.1	Généralités	45

7.2	Composants	45
7.2.1	Fichiers communs	45
7.2.1.1	Fichier /vitam/conf/<composant>/sysconfig/java_opts	45
7.2.1.2	Fichier /vitam/conf/<composant>/logback.xml	46
7.2.1.3	Fichier /vitam/conf/<composant>/logback-access.xml	47
7.2.1.4	Fichier /vitam/conf/<composant>/jetty-config.xml	49
7.2.1.5	Fichier /vitam/conf/<composant>/logbook-client.conf	54
7.2.1.6	Fichier /vitam/conf/<composant>/server-identity.conf	54
7.2.1.7	Fichier /vitam/conf/<composant>/antisamy-esapi.xml	55
7.2.1.8	Fichier /vitam/conf/<composant>/vitam.conf	68
7.2.1.9	Fichier /vitam/conf/<composant>/vitam.metrics.conf	69
7.2.1.10	Fichier /vitam/conf/<composant>/java.security	70
7.2.2	access external	70
7.2.2.1	Présentation	70
7.2.2.2	Configuration / fichiers utiles	70
7.2.2.2.1	Fichier access-external.conf	70
7.2.2.2.2	Fichier access-internal-client.conf	70
7.2.2.2.3	Fichier functional-administration-client.conf	70
7.2.2.3	Opérations	71
7.2.3	access-internal	71
7.2.3.1	Présentation du composant	71
7.2.3.2	Configuration / fichiers utiles	71
7.2.3.2.1	Fichier access.conf	71
7.2.3.2.2	Fichier storage-client.conf	72
7.2.3.2.3	Fichier metadata-client.conf	72
7.2.3.3	Opérations	72
7.2.4	Cerebro	73
7.2.4.1	Présentation	73
7.2.4.2	Configuration / fichiers utiles	73
7.2.4.2.1	Fichier /vitam/conf/cerebro/application.conf	73
7.2.4.3	Opérations	74
7.2.5	common-plugin	75
7.2.5.1	Présentation du composant	75
7.2.5.2	Classes utiles	75
7.2.5.2.1	Classe Item Status	75
7.2.5.2.2	Classe VitamAutoCloseable	75
7.2.5.2.3	Classe ParameterHelper	75
7.2.5.2.4	Classe VitamParameter	75
7.2.5.2.5	Classe ProcessingException	75
7.2.5.2.6	Classe IOParameter	75
7.2.5.2.7	Classe ProcessingUri	76
7.2.5.2.8	Classe UriPrefix	76
7.2.5.2.9	Classe AbstractWorkerParameters	76
7.2.5.2.10	Classe DefaultWorkerParameters	76
7.2.5.2.11	Classe WorkerParameterName	76
7.2.5.2.12	Classe WorkerParameters	76
7.2.5.2.13	Classe WorkerParametersDeserializer	76
7.2.5.2.14	Classe WorkerParametersFactory	76
7.2.5.2.15	Classe WorkerParametersSerializer	76
7.2.5.2.16	Interface HandlerIO	76
7.2.5.2.17	Classe WorkerAction	77
7.2.5.2.18	Classe HandlerIOImpl	77
7.2.6	consul	77
7.2.6.1	Présentation	77

7.2.6.1.1	Cas serveur	77
7.2.6.1.2	Cas agent	77
7.2.6.2	Configuration / fichiers utiles	77
7.2.6.2.1	Cas des applicatifs monitorés par Consul	78
7.2.6.2.1.1	Fichier /vitam/conf/consul/service-<composant>.json	78
7.2.6.3	Opérations	79
7.2.7	Kibana interceptor	79
7.2.7.1	Présentation	79
7.2.7.2	Configuration / fichiers utiles	80
7.2.7.2.1	Fichier elastic-kibana-interceptor.conf	80
7.2.7.3	Opérations	80
7.2.8	elasticsearch chaîne de log	81
7.2.8.1	Présentation	81
7.2.8.2	Configuration / fichiers utiles	81
7.2.8.2.1	Fichier log4j2.properties	81
7.2.8.2.2	Fichier elasticsearch.yml	83
7.2.8.2.3	Fichier sysconfig/elasticsearch	85
7.2.8.2.4	Fichier /usr/lib/tmpfiles.d/elasticsearch-data.conf	87
7.2.8.3	Opérations	87
7.2.9	elasticsearch Vitam	88
7.2.9.1	Présentation	88
7.2.9.2	Configuration / fichiers utiles	88
7.2.9.2.1	Fichier log4j2.properties	88
7.2.9.2.2	Fichier elasticsearch.yml	90
7.2.9.2.3	Fichier sysconfig/elasticsearch	93
7.2.9.2.4	Fichier /usr/lib/tmpfiles.d/elasticsearch-data.conf	95
7.2.9.3	Opérations	95
7.2.10	functional administration	95
7.2.10.1	Présentation	95
7.2.10.2	Configuration / fichiers utiles	95
7.2.10.2.1	Fichier functional-administration.conf	96
7.2.10.2.2	Passage des identifiants des référentiels en mode esclave	96
7.2.10.3	Opérations	98
7.2.11	ihm-demo	98
7.2.11.1	Présentation	98
7.2.11.2	Configuration / fichiers utiles	98
7.2.11.2.1	Fichier access-external-client.conf	99
7.2.11.2.2	Fichier ihm-demo.conf	99
7.2.11.2.3	Fichier ingest-external-client.conf	99
7.2.11.2.4	Fichier shiro.ini	99
7.2.11.3	Configuration de apache shiro	102
7.2.11.4	Présentation authentification via LDAP et via certificat	102
7.2.11.5	Décryptage de shiro.ini	102
7.2.11.6	Opérations	103
7.2.12	ihm-recette	104
7.2.12.1	Présentation	104
7.2.12.2	Configuration / fichiers utiles	104
7.2.12.2.1	Fichier access-external-client.conf	104
7.2.12.2.2	Fichier ihm-recette.conf	104
7.2.12.2.3	Fichier ihm-recette-client.conf	105
7.2.12.2.4	Fichier ingest-external-client.conf	106
7.2.12.2.5	Fichier functional-administration-client.conf	106
7.2.12.2.6	Fichier shiro.ini	106

7.2.12.2.7	Fichier <code>storage-client.conf</code>	107
7.2.12.2.8	Fichier <code>storage-offer.conf</code>	107
7.2.12.2.9	Fichier <code>tnr.conf</code>	108
7.2.12.3	Opérations	108
7.2.13	<code>ingest-external</code>	109
7.2.13.1	Présentation	109
7.2.13.2	Configuration / fichiers utiles	109
7.2.13.2.1	Fichier <code>ingest-external.conf</code>	109
7.2.13.2.2	Fichier <code>ingest-internal-client.conf</code>	109
7.2.13.3	Opérations	110
7.2.14	<code>ingest-internal</code>	110
7.2.14.1	Présentation	110
7.2.14.2	Configuration / fichiers utiles	110
7.2.14.2.1	Fichier <code>ingest-internal.conf</code>	111
7.2.14.3	Opérations	111
7.2.15	<code>log server</code>	111
7.2.15.1	Présentation	111
7.2.15.2	Configuration / fichiers utiles	111
7.2.15.3	Opérations	112
7.2.16	<code>logbook</code>	112
7.2.16.1	Présentation	112
7.2.16.2	Configuration / fichiers utiles	112
7.2.16.2.1	Fichier <code>logbook.conf</code>	113
7.2.16.3	Opérations	114
7.2.17	<code>metadata</code>	114
7.2.17.1	Présentation	114
7.2.17.2	Configuration / fichiers utiles	115
7.2.17.2.1	Fichier <code>metadata.conf</code>	115
7.2.17.3	Opérations	115
7.2.18	<code>mongoC</code>	116
7.2.18.1	Présentation	116
7.2.18.2	Configuration / fichiers utiles	116
7.2.18.3	Opérations	116
7.2.19	<code>mongoD</code>	117
7.2.19.1	Présentation	117
7.2.19.2	Configuration / fichiers utiles	117
7.2.19.3	Opérations	117
7.2.20	<code>mongoS</code>	117
7.2.20.1	Présentation	117
7.2.20.2	Configuration / fichiers utiles	117
7.2.20.3	Opérations	118
7.2.21	<code>processing</code>	118
7.2.21.1	Présentation	118
7.2.21.2	Configuration / fichiers utiles	118
7.2.21.2.1	Fichier <code>processing.conf</code>	119
7.2.21.2.2	Fichier <code>version.conf</code>	119
7.2.21.2.3	Fichier <code>storage-client.conf</code>	119
7.2.21.3	Opérations	119
7.2.22	<code>Security internal</code>	120
7.2.22.1	Présentation	120
7.2.22.2	Configuration / fichiers utiles	120
7.2.22.2.1	Fichier <code>security-internal.conf</code>	120
7.2.22.3	Opérations	120
7.2.23	<code>siegfried</code>	121

7.2.23.1	Présentation	121
7.2.23.2	Configuration / fichiers utiles	121
7.2.23.3	Opérations	121
7.2.24	storage-engine	122
7.2.24.1	Présentation	122
7.2.24.2	Configuration / fichiers utiles	122
7.2.24.2.1	Fichier driver-location.conf	122
7.2.24.2.2	Fichier driver-mapping.conf	122
7.2.24.2.3	Fichier static-offer.json	122
7.2.24.2.4	Fichier static-strategy.json	123
7.2.24.2.5	Fichier storage-engine.conf	123
7.2.24.3	Opérations	124
7.2.25	offer	124
7.2.25.1	Présentation	124
7.2.25.2	Configuration / fichiers utiles	124
7.2.25.2.1	Fichier default-offer.conf	125
7.2.25.2.2	Fichier default-storage.conf	125
7.2.25.3	Opérations	125
7.2.26	worker	126
7.2.26.1	Présentation	126
7.2.26.2	Configuration / fichiers utiles	126
7.2.26.2.1	Fichier format-identifiants.conf	126
7.2.26.2.2	Fichier functional-administration-client.conf.j2	126
7.2.26.2.3	Fichier metadata-client.conf	127
7.2.26.2.4	Fichier storage-client.conf	127
7.2.26.2.5	Fichier version.conf	127
7.2.26.2.6	Fichier worker.conf	127
7.2.26.3	Opérations	128
7.2.27	workspace	128
7.2.27.1	Présentation	128
7.2.27.2	Configuration / fichiers utiles	128
7.2.27.2.1	Fichier workspace.conf	129
7.2.27.3	Opérations	129
8	Intégration d'une application externe dans Vitam	130
8.1	Prérequis	130
8.2	Intégration de certificats clients de VITAM	130
8.2.1	Authentification applicative SIA	130
8.2.2	Authentification <i>personae</i>	131
8.2.2.1	Ajout d'un certificat pour l'authentification Personae	131
8.2.2.2	Suppression d'un certificat pour l'authentification Personae	131
8.3	Déploiement des keystores	131
8.3.1	Vitam n'est pas encore déployé	131
8.3.2	Vitam est déjà déployé	131
9	Aide à l'exploitation	132
9.1	Analyse de premier niveau	132
9.1.1	Etat par Consul	132
9.1.2	Etat par Kibana	133
9.2	Playbook ansible pour échanger avec le support	133
10	Questions Fréquemment Posées	135
10.1	Présentation	135
10.2	Retour d'expérience / cas rencontrés	135

10.2.1	Crash rsyslog, code killed, signal : BUS	135
10.2.2	Mongo-express ne se connecte pas à la base de données associée	135
10.2.3	Elasticsearch possède des shard non alloués (état « UNASSIGNED »)	135
10.2.4	Elasticsearch possède des shards non initialisés (état « INITIALIZING »)	136
10.2.5	MongoDB semble lent	136
10.2.6	Les shards de MongoDB semblent mal équilibrés	137
10.2.7	L'importation initiale (profil de sécurité, certificats) retourne une erreur	138
10.2.8	Problème d'ingest et/ou d'access	138
11	Exploitation par composant	139
11.1	Access	139
11.1.1	Introduction	139
11.2	Batch-Report	139
11.2.1	Présentation	139
11.2.2	batch-report-exploitation	140
11.3	Common	140
11.3.1	Présentation	140
11.3.2	Format Identifièrs	140
11.3.2.1	Configuration des services d'identification des formats	140
11.4	Functional administration	141
11.4.1	Présentation	141
11.4.2	Functional administration	141
11.4.2.1	Configuration du Functional administration	141
11.5	Ingest	141
11.5.1	Introduction	141
11.5.2	ingest-external-exploitation	141
11.5.3	ingest-internal-exploitation	142
11.6	Security-Internal	142
11.6.1	Introduction	142
11.6.2	security-internal-exploitation	142
11.7	Logbook	143
11.7.1	Présentation	143
11.7.2	Logbook Exploitation	143
11.7.2.1	Configuration du Logbook	143
11.8	Metadata	143
11.8.1	Présentation	143
11.9	Processing	143
11.9.1	Introduction	143
11.9.1.1	But de cette documentation	143
11.9.2	Processing	144
11.9.2.1	Configuration du worker	144
11.9.2.2	Supervision du service	144
11.10	Storage	144
11.10.1	Introduction	144
11.10.1.1	But de cette documentation	144
11.10.2	Storage Engine	144
11.10.2.1	Configuration du moteur de stockage	145
11.10.2.2	Configuration du driver de l'offre de stockage par défaut	146
11.10.2.3	Supervision du service	147
11.10.3	Storage Offer Default	147
11.10.3.1	Configuration de l'offre de stockage	147
11.10.3.2	Supervision du service	147
11.11	Technical administration	147
11.11.1	Présentation	147

11.12 Worker	147
11.12.1 Introduction	147
11.12.1.1 But de cette documentation	147
11.12.2 Storage Engine	147
11.12.2.1 Configuration du worker	148
11.12.2.2 Supervision du service	148
11.13 Workspace	148
11.13.1 Présentation	148
12 Annexes	149
12.1 Cycle de vie des certificats	149
12.2 Gestion des anomalies en production	151
12.2.1 Numérotation des versions	151
12.2.2 Mise à disposition du logiciel	151
12.2.3 Gestion des patches	151
Index	155

CHAPITRE 1

Introduction

1.1 But de cette documentation

Ce document a pour but de permettre de fournir à une équipe d'exploitants de 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.

2.1 Information concernant les licences

La solution logicielle *VITAM* est publiée sous la licence [CeCILL 2.1](http://www.cecill.info/licences/Licence_CeCILL_V2.1-fr.html)¹ ; la documentation associée (comprenant le présent document) est publiée sous [Licence Ouverte V2.0](https://www.etalab.gouv.fr/wp-content/uploads/2017/04/ETALAB-Licence-Ouverte-v2.0.pdf)².

2.2 Documents de référence

2.2.1 Documents internes

Tableau 1 – Documents de référence VITAM

Nom	Lien
<i>DAT</i>	http://www.programmevitam.fr/ressources/DocCourante/html/archi
<i>DIN</i>	http://www.programmevitam.fr/ressources/DocCourante/html/installation
<i>DEX</i>	http://www.programmevitam.fr/ressources/DocCourante/html/exploitation
Release notes	

2.2.2 Référentiels externes

2.3 Glossaire

API Application Programming Interface

BDD Base De Données

CA Certificate Authority

¹http://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>

- COTS** Component Off The Shelves ; 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.
- DAT** Dossier d'Architecture Technique
- DEX** Dossier d'EXploitation
- DIN** Dossier d'Installation
- 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
- DUA** Durée d'Utilité Administrative
- IHM** Interface Homme Machine
- 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*
- 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
- NoSQL** Base de données non-basée sur un paradigme classique des bases relationnelles. [Définition NoSQL](#)⁵
- 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.
- 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
- 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
- SIA** Système d'Informations Archivistique
- SIP** Submission Information Package
- TNR** Tests de Non-Régression
- VITAM** Valeurs Immatérielles Transférées aux Archives pour Mémoire

https://fr.wikipedia.org/wiki/Domain_Name_System_Security_Extensions

https://fr.wikipedia.org/wiki/Attaque_de_l'homme_du_milieu

<https://fr.wikipedia.org/wiki/NoSQL>

https://fr.wikipedia.org/wiki/Infrastructure_%C3%A0_cl%C3%A9s_publicques

https://fr.wikipedia.org/wiki/Representational_state_transfer

CHAPITRE 3

Expertises requises

Les équipes en charge du déploiement et de l'exploitation de la solution 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 *VITAM* :

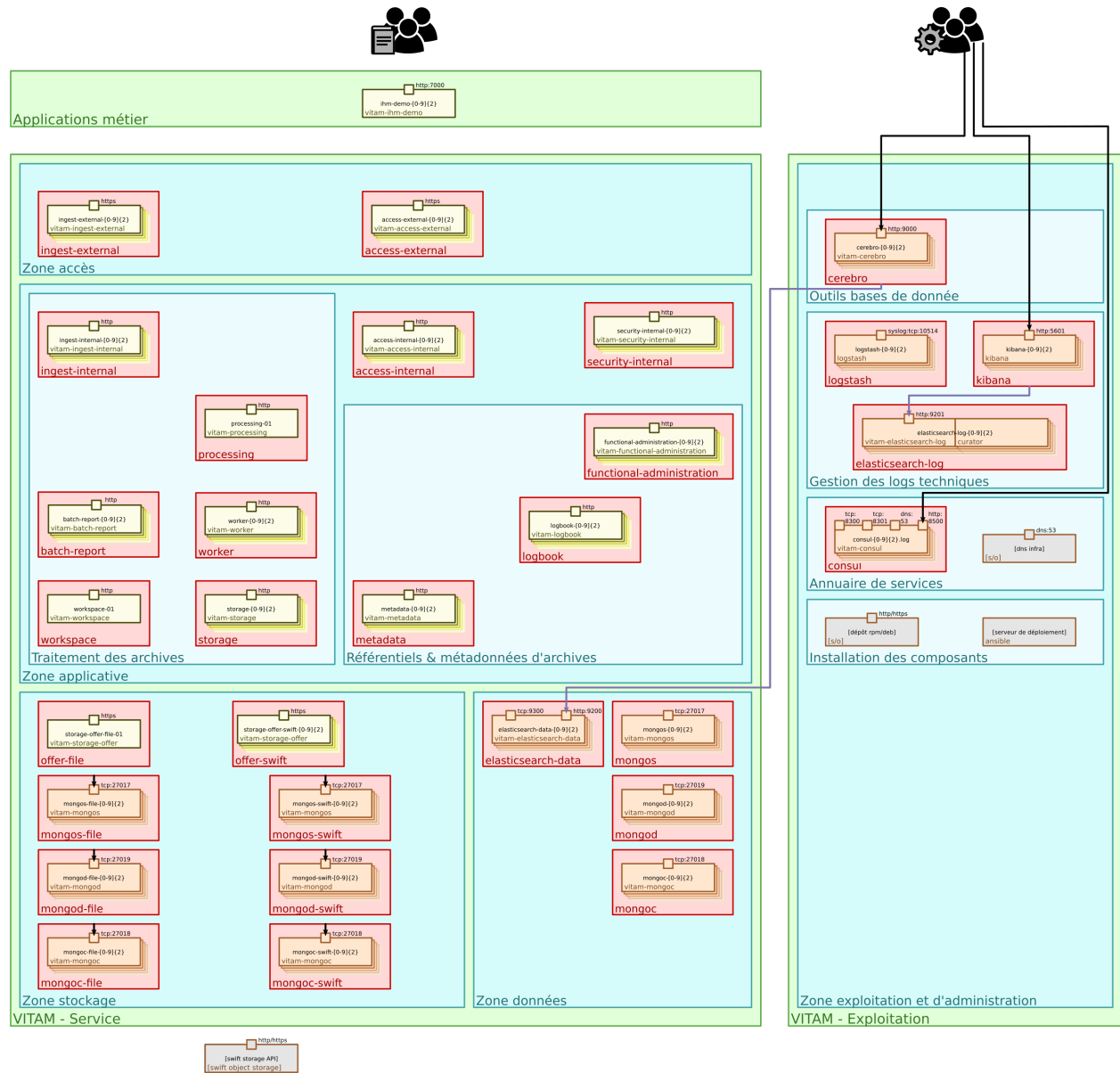


Fig. 1 – Vue d'ensemble d'un déploiement *VITAM* : zones, composants

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.

5.1 Gestion des accès

1. API

La gestion des accès à l'API se fait via les *granted stores* (cf. *Fichiers communs* (page 45) pour les fichiers communs, cf. *Configuration / fichiers utiles* (page 70) pour access-external et *Configuration / fichiers utiles* (page 110) pour ingest-external).

2. IHM d'admin

Dans cette version, la gestion des utilisateurs se fait par configuration d'un fichier plat (cf. *Opérations* (page 103)).

5.2 Portails d'administration

5.2.1 Technique

Aucun portail d'administration technique n'est prévu dans cette version de VITAM.

5.2.2 Fonctionnel

Le portail d'administration fonctionnel est intégré à l'IHM Démo dans cette version de VITAM (cf. *Présentation* (page 98)).

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*

Prudence : limitation technique à ce jour ; il n'est pas possible de définir des variables JVM différentes pour des composants colocalisés sur une même partition.

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 régénérer les stores (dans `environments/keystores`) et lancer leur redéploiement via cette commande ansible :

```
# 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
# 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 vitam

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

Ces documents détaillent les pré-requis, la configuration des fichiers pour effectuer un déploiement. Le *DIN* explique également comment valider une montée de version applicative de *VITAM*.

Voir aussi :

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

5.4.3 Ajouter un/des instances de services VITAM

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

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.

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)>`

5.5 Interruption / maintenance

5.5.1 Procédure d'arrêt

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.

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

Avertissement : Dans la version actuelle, il est fortement recommandé de positionner les traitements courants en pause avant de lancer la procédure d'arrêt.

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

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.

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 » intégrés dans la solution logicielle *VITAM*.

5.5.4 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 `ingest-external` et `access-external`
- fichier de *playbook* `start_external.yml`, permettant de démarrer sélectivement `ingest-external` et `access-external`

Ils permettent d'empêcher l'usage de *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 *VITAM*.

5.5.5 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*

5.5.6 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 chaîne 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 chaîne de log (logstash / cluster elasticsearch log / kibana-log), cerebro et les docker mongo-express et elasticsearch-head

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

5.5.7 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*

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

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'ansible 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*.

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://elasticsearch-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://elasticsearch-data.service.${consul_domain}:9200/_snapshot/vitam_
↳backup/_verify
```

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

```
curl -X PUT http://elasticsearch-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 copie 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 mongodbl.example.net --port 3017 --username vitamdb-admin --password "pass"
${output_dir}/${fichier}
```

Note : Se reporter aux fichiers associés à l'ansible 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://elasticsearch-data.service.${consul_domain}:9200/_snapshot/vitam_
↳backup/
```

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

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

5.7 Batches et traitements

5.7.1 Curator

Il existe des jobs Curator de :

- fermeture d'index
- suppression d'index fermés

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

5.7.2 Sécurisation des journaux d'opérations

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

- *vitam-traceability-operations* (page 41)

5.7.3 Sécurisation des cycles de vie

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

- *vitam-traceability-lfc-unit* (page 41)
- *vitam-traceability-lfc-objectgroup* (page 42)

5.7.4 Sécurisation des offres de stockages

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

- *vitam-storage-log-backup* (page 41)
- *vitam-storage-log-traceability* (page 41)

5.8 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) et les stocker dans l'offre de stockage.

Note : Log shipping

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

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

5.8.1 Déclenchement

La sauvegarde des données graphe est lancée via un timer systemd (*vitam-metadata-store-graph* (page 44)) qui démarre un service systemd concerné.

- Le timer se lance chaque 30 minutes (modifiable selon le besoin)
- Le sauvegarde des données graphe se fait sur une intervalle de temps (Depuis la dernière sauvegarde jusqu'à temps présent)
- Le fichier généré est un fichier 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 container en plus dans l'offre de stockage)

5.8.1.1 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 :

La *Reconstruction* (page 16) des métadonnées pour avoir plus d'information sur la reconstruction.

5.9 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.

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.

5.9.1 Déclenchement

Le recalcul de graphe est déclenché par l'appel au point d'API porté par l'URL suivante : `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) :

```
{
  "$roots": [
    "aeaqaaaaaqhdytymabdeialenehzphiaaaeq",
    "aeaqaaaaaqhdytymabdeialenehzpbyaaaajq"
  ],
  "guid_n": [
  ],
  "$query": [],
  "$projection": {}
}
```

La valeur utilisée pour la basic authentication sont issues du fichier vault.yml 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 ES sont organisés par tenant.

5.10 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.

5.10.1 Déclenchement

5.10.1.1 Cas du site primaire :

La reconstruction se réalise de la manière suivante :

- Arrêt de *VITAM* sur le site à reconstruire
 - 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
- Purge des données (le cas échéant) stockées dans MongoDB-data
- Purge des données (le cas échéant) stockées dans Elasticsearch-data,
- Reconfiguration et démarrage en tant que site secondaire :
 - Paramétrer la variable `primary_site` à `false` puis utiliser le playbook `ansible-vitam/vitam.yml`
- Dès lors, l'accès utilisateur reste coupé, et l'intégralité des données est reconstruit 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 »
- Restaurer la base `identity` de MongoDB-data (utiliser par exemple l'utilitaire commande `mongorestore`⁸ permettant de restaurer des données à partir d'une sauvegarde de base de données créée par le programme `mongodump`⁹ vers une instance MongoDB)
- 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`

5.10.1.2 Cas du site secondaire :

La reconstruction se réalise de la manière suivante :

- Arrêt de *VITAM* sur le site à reconstruire
 - 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
- Purge des données (le cas échéant) stockées dans MongoDB-data

⁸<https://docs.mongodb.com/manual/reference/program/mongorestore>
⁹<https://docs.mongodb.com/manual/reference/program/mongodump>

- Purge des données (le cas échéant) stockées dans Elasticsearch-data,
- 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

5.11 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 s'appuie sur les procédures décrites dans les chapitres suivants :

- Resynchronisation d'une offre par rapport à une autre en cas de la perte totale et définitive d'une offre (*Resynchronisation d'une offre* (page 19))
- 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 16))

5.11.1 Déclenchement

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

Le déclenchement du PRA s'effectue selon la procédure suivante :

- 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
- 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
- Restaurer la base identity de MongoDB-data (utiliser par exemple l'utilitaire commande `mongorestore`¹⁰ permettant de restaurer des données à partir d'une sauvegarde de base de données créée par le programme `mongodump`¹¹ vers une instance MongoDB)
- Reconfigurer le site secondaire en site primaire :

¹⁰<https://docs.mongodb.com/manual/reference/program/mongorestore>
¹¹<https://docs.mongodb.com/manual/reference/program/mongodump>

- 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 (*Décommissionnement d'une offre* (page 19))
 - 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 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 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.

5.11.2 Retour en situation nominale

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

- Rétablissement du contenu du site primaire initial par reconfiguration temporaire en tant que site secondaire
- Retour à la configuration bi-site initiale

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

5.11.3 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'effectue 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`
- 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)
- Restaurer la base `identity` de `MongoDB-data` (utiliser par exemple l'utilitaire commande `mongorestore`¹² permettant de restaurer des données à partir d'une sauvegarde de base de données créée par le programme `mongodump`¹³ vers une instance `MongoDB`)
- Resynchroniser les offres de stockage à partir des offres du site de secours en se référant à la procédure suivante *Resynchronisation totale d'une offre* (page 22)

¹²<https://docs.mongodb.com/manual/reference/program/mongorestore>
¹³<https://docs.mongodb.com/manual/reference/program/mongodump>

- 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 reconstruction 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 bi-site (primaire/secondaire). Le service est de nouveau disponible sur le site primaire initial.

5.12 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.

Le mécanisme de resynchronisation d'une offre par rapport à une autre nécessite une intervention manuelle permettant de remédier à la perte de données dans le système en proposant différentes stratégies de resynchronisation :

- La resynchronisation totale d'une offre de stockage (cas de la perte totale et définitive d'une offre)
- La resynchronisation partielle d'une offre de stockage (cas de l'indisponibilité sur un temps plus ou moins long d'une offre)

Le processus d'entrée d'un *SIP* n'étant réussi que si et seulement si toutes les offres de stockage définies dans la stratégie sont accessibles, et que tous les fichiers sont bien copiés sur la totalité de ces offres, il sera nécessaire de décommissionner l'offre afin de permettre à nouveau les entrées de *SIP* (ingest/versement).

Note : Dans cette version, la resynchronisation partielle d'une offre de stockage n'est pas supportée.

5.12.1 Décommissionnement d'une offre

Lors de la détection de la perte d'une offre de stockage, afin de permettre à nouveau les versements, l'offre pourra être décommissionnée en fonction du mode d'utilisation choisi :

- Mode « lecture/écriture » : la stratégie de stockage doit être modifiée

- Mode « lecture seule » (recherche et consultation avec profil de droits dédié) : la stratégie de stockage ne change pas

Le décommissionnement d'une offre s'effectue selon la procédure suivante :

- Editer le fichier de configuration de la stratégie de stockage `offer_opts.yml` afin de retirer l'offre posant problème (commenter les lignes correspondantes) :

```
vitam_strategy:
- name: <offer-x>
  referent: true
# <offer-y> is down
#- name: <offer-y>
#   referent: false

vitam_offers:
  <offer-x>:
    provider: filesystem
# <offer-y> is down
#<offer-y>:
#   provider: filesystem
```

- 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)

5.12.2 Ajout d'une 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
- 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 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 stockage 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

```

5.12.3 Resynchronisation totale d'une offre

Suite à l'ajout d'une offre de stockage, celle-ci doit être resynchronisée vis-à-vis d'une offre source selon 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 http://<offer-x.hosts-storage-
↪offer-default>:29102/storage/v1/offerSync < query
{
  "offerSource": "<offer-x>.service.consul",
  "offerDestination": "<offer-z>.service.consul"
}
```

- 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 `offerSource` correspond à l'id de l'offre source utilisée pour la resynchronisation de la nouvelle offre
- Le paramètre `offerDestination` correspond à l'id de l'offre à resynchroniser
- Suivre le déroulement de la resynchronisation dans les logs du composant `storage offer` avec la commande suivante :

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

- Les différents messages « *[OfferSync] : successful synchronization of category : <container>, tenant : <tenantId>, offset : <offset>* » indiquent la fin de resynchronisation de la catégorie `<container>`, pour le tenant `<tenantId>`, avec l'offset de resynchronisation `<offset>`
- Le message « *The offers" synchronization is completed* » indique la fin du processus de resynchronisation
- En cas d'interruption ou d'échec de la resynchronisation (le message indiquant la fin du processus de resynchronisation n'est pas affiché), il est possible de relancer le processus de la manière suivante :

```
curl -v -X POST -u adminUser:adminPassword http://<hosts-storage-offer-
↪default>:29102/storage/v1/offerSync < query
{
  "offerSource": "<offer-x>.<consul_domain>",
  "offerDestination": "<offer-z>.<consul_domain>",
  "offset": <offset>,
  "containerToSync": <container>,
  "tenantIdToSync": <tenantId>
}
```

Dans le log de la dernière exécution du processus de resynchronisation : « *[OfferSync] : successful synchronization of category : <container>, tenant : <tenantId>, offset : <offset>* »

- Le paramètre `offset` correspond à la valeur `<offset>` observée
- Le paramètre `containerToSync` correspond à la valeur `<container>` observée
- Le paramètre `tenantIdToSync` correspond à la valeur `<tenantId>` observée

5.13 Procédure d'exploitation suite a 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.

5.13.1 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_0, unit_1 etc ...)

```
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"
```

5.13.2 Changement de type d'une ontologie existante

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

On récupère d'abord l'ancien index

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

On crée 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 crée 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"
```

(suite sur la page suivante)

(suite de la page précédente)

```

    },
    "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'
```

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

Avvertissement : 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.

5.14 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 à leur réception (toutes ou seulement d'un type donnée, sur tout 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'opération, sur l'ensemble des tenants ou sur un tenant donné seulement.

5.14.1 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_admin }}:{{ 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_admin }}:{{ vitam.metadata.port_admin }}/admin-external/v1/
↪forcepause`
-H 'X-Tenant-Id: 0'
-H 'X-Access-Contract-Id: ContratTNR'
-H 'Content-Type: application/json;charset=UTF-8'

```

(suite sur la page suivante)

(suite de la page précédente)

```
-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}'
```

5.14.2 Sortie de la mise en pause forcée

La sortie 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_admin }}:{{ 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_admin }}:{{ vitam.metadata.port_admin }}/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 seront perdus.

5.15 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 130).

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

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

- Le système *VITAM* récupère le CRL d'un CA qui a émis le certificat présent dans *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)

- 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 Le système *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.

5.15.1 Procédure de révocation

La prise en compte d'une CRL par le système *VITAM* s'effectue via la commande suivante :

```
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
```

Note : Un retour arrière est possible en modifiant le statut du certificat directement dans la base MongoDB (collection `identity.Certificate`), en passant le champ "Status" de "REVOKED" à "VALID".

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 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.

6.2.1.2 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.

6.2.1.3 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 Kibana¹⁴.

L'application Kibana comporte 4 sections qui seront développées :

- **Discover**
- **Visualize**

<https://www.elastic.co/fr/products/kibana>

- 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.

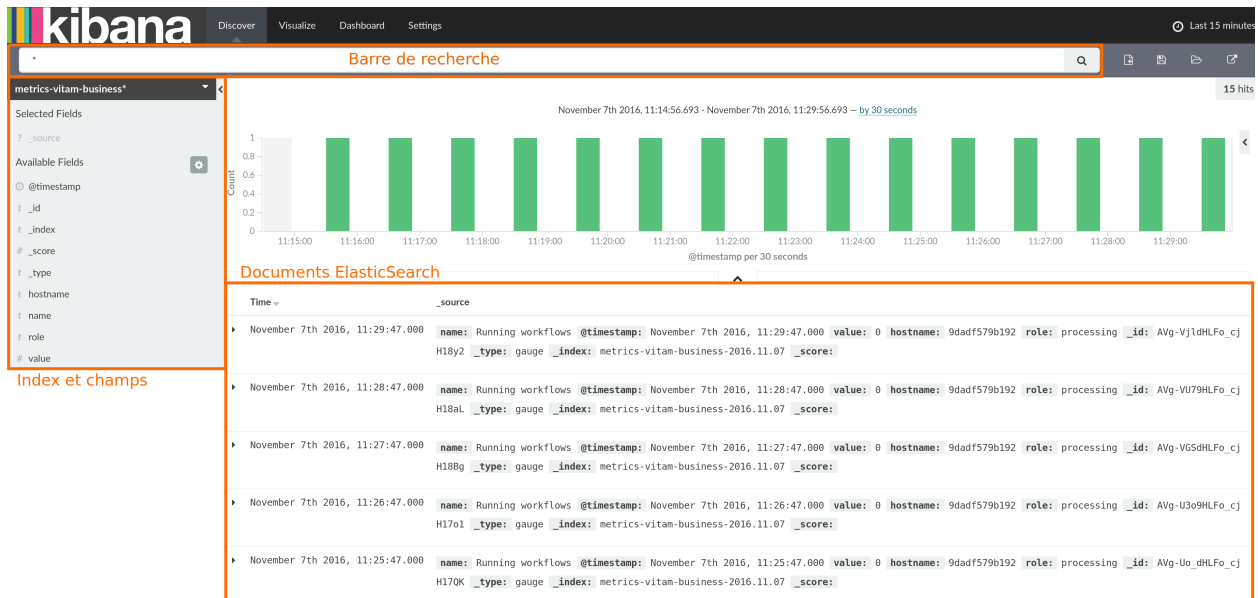
Voir aussi :

[Documentation officielle de Kibana](#) ¹⁵

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 sélectionner un index dans la barre latérale gauche, de choisir les champs que l'on souhaite consulter (optionnel) et les données apparaissent 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 exécutée, 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 :

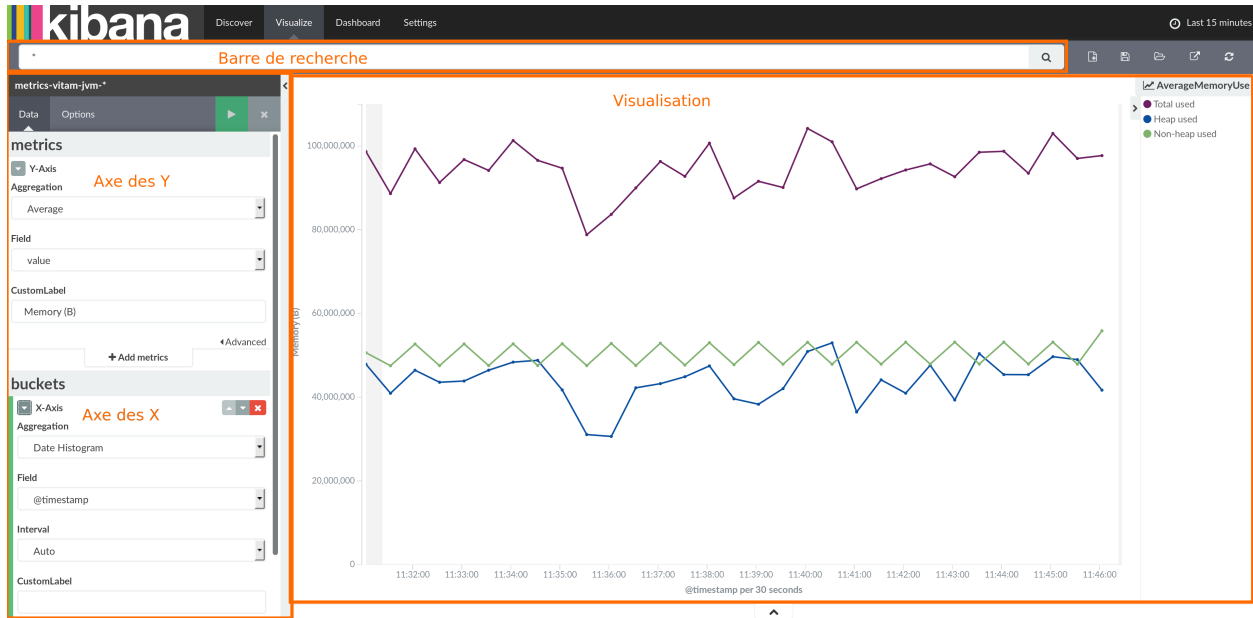
- **Area chart** : utile pour un regroupement de séries chronologiques dans lequel le total des séries est plus important que la différence entre plusieurs séries.
- **Data table** : un tableau de données classique.
- **Line chart** : graphique pour des séries temporelles. Très utile pour comparer deux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

<https://www.elastic.co/guide/en/kibana/current/index.html>

- **Tile map** : représentation de coordonnées géographiques sur une carte.
- **Vertical bar chart** : un histogramme classique.

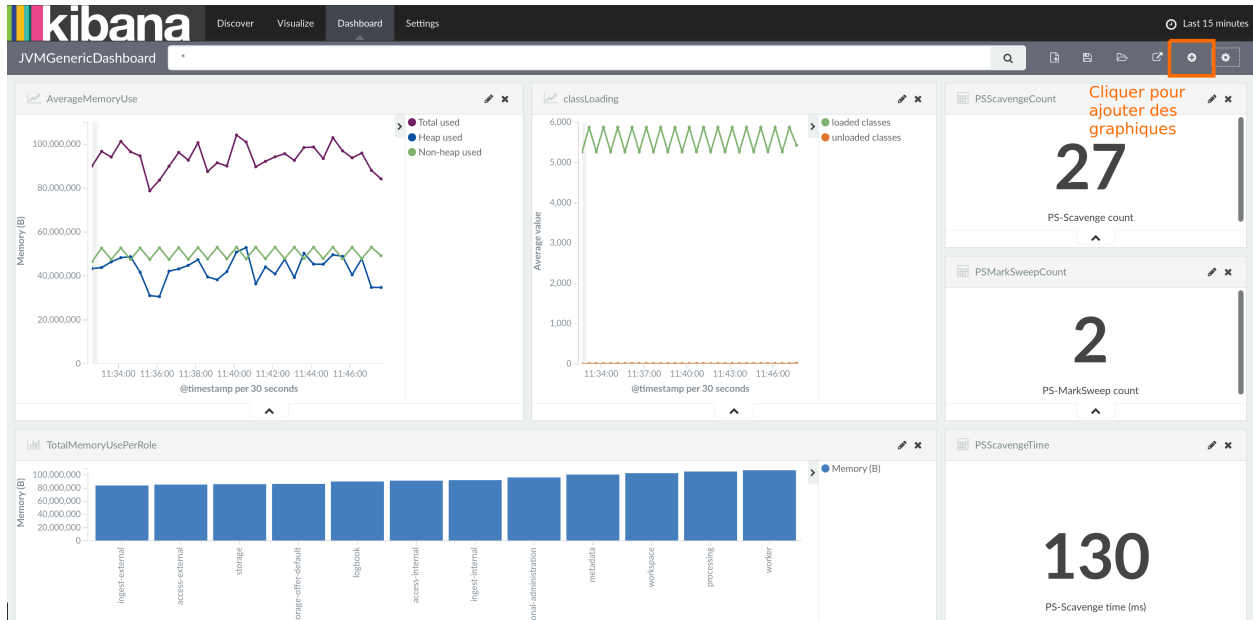
La barre latérale gauche du panneau de visualisation permet de configurer la donnée à représenter. Pour l'axe des Y, il est impératif d'utiliser un agrégation (moyenne, minimum/maximum, écart type. . .) sur une valeur pour la représenter. En fonction du graphique sélectionné, il est possible de configurer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

La solution logicielle :term‘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 ressources requises par le composant et renvoyant un statut d'état de ces ressources.

6.3.1 Détail

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": {
    "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 servi 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 :

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

Signification des champs :

- **statusCode** : 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** :
 - statusCode : 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

6.4 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 :term'VITAM'.

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

- `rsyslog` 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/<service_id>/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'appendeur `RollingFileAppender`¹⁶, 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`)

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

- 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'appendeur `RollingFileAppender`¹⁷, 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.

6.5 Audit

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

6.6 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.

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

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 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.

6.9.2 Appels REST

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

Il suffit juste de lancer un appel curl sur l'accès externe :

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

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

6.9.3 Worklow en FATAL

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

6.9.3.1 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 *Workflow*.

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

Si le composant Logbook est défectueux et ne répond plus, alors un FATAL pourra être obtenu pour tous les Handlers, mais pou

- `CommitLifecycleActionHandler`
- `CommitLifecycleObjectGroupActionHandler`
- `CommitLifecycleUnitActionHandler`
- `ListLifecycleTraceabilityActionHandler`

- FinalizeLifecycleTraceabilityActionHandler
- RollBackActionHandler

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

- CheckArchiveProfileRelationActionHandler
- CheckArchiveProfileActionHandler
- GenerateAuditReportActionHandler
- PrepareAuditActionHandler

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

- CheckStorageAvailabilityActionHandler
- FinalizeLifecycleTraceabilityActionHandler
- GenerateAuditReportActionHandler
- PrepareTraceabilityCheckProcessActionHandler
- PutBinaryOnWorkspace
- CheckIntegrityObjectPlugin
- CheckExistenceObjectPlugin
- StoreMeta-dataObjectGroupActionPlugin
- StoreMeta-dataUnitActionPlugin
- StoreObjectActionHandler
- StoreObjectGroupActionPlugin

Si le composant ProcessingManagement est défectueux et 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

6.9.3.2 Distributor

Plusieurs cas peuvent provoquer un FATAL au niveau du processing :

- si Metadata ou Workspace est down
- si un Handler (ou plugin) inexistant est appelé.
- si le distributeur tente d'appeler une famille de worker inexistante

6.9.3.3 Processing - State Machine

Dans le cas où 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.

6.9.4 Redémarrer un processus en cas de pause

6.9.4.1 Trouver la cause

De manière générale, il convient d'identifier le composant (ou les composants) posant problème. Il s'agira majoritairement de Metadata, de Logbook, du Storage 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.

6.9.4.2 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.

6.9.4.2.1 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).

6.9.4.2.2 Rejouer une étape

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

Cette action aura pour résultat d'exécuter une 2ème fois l'étape qui a échoué. En sortie de ce replay, normalement, le statut du workflow doit passer à OK et l'état à PAUSE.

6.9.4.2.3 Prochaine étape

Depuis l'*IHM*, exécuter l'étape suivante en cliquant sur l'icône « Next ». Via les *API*, il suffit juste 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, normalement, le statut du workflow doit passer à OK et l'état à PAUSE.

6.9.4.2.4 Finaliser le workflow

Une fois sûrs de notre coup, il est maintenant 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 juste 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, uniquement les TNR (Tests de Non Régression) 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 branche

- 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, alors un code 500 sera obtenu.

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 événements 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.

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-site.

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-site.

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 (donc : 0h00, 1h00, ...)

6.11.2.2 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 (donc : 0h10, 1h10, ...)

6.11.2.3 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.

6.11.2.4 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.

6.11.2.5 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.

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-site.

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.

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.

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.

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-site.

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.

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.

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.

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.

Exploitation des composants 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 Composants

7.2.1 Fichiers communs

Les composants de la solution logicielle *VITAM* utilisent un socle de fichiers communs.

7.2.1.1 Fichier `/vitam/conf/<composant>/sysconfig/java_opts`

Ce fichier définit les JVMARGS.

```
1 #*****
2 # Copyright French Prime minister Office/SGMAP/DINSIC/Vitam Program (2015-2019)
3 #
4 # contact.vitam@culture.gouv.fr
5 #
6 # This software is a computer program whose purpose is to implement a digital_
  ↳ archiving back-office system managing
```

(suite sur la page suivante)

```

7  # high volumetry securely and efficiently.
8  #
9  # This software is governed by the CeCILL 2.1 license under French law and abiding by
↳the rules of distribution of free
10 # software. You can use, modify and/ or redistribute the software under the terms of
↳the CeCILL 2.1 license as
11 # circulated by CEA, CNRS and INRIA at the following URL "http://www.cecill.info".
12 #
13 # As a counterpart to the access to the source code and rights to copy, modify and
↳redistribute granted by the license,
14 # users are provided only with a limited warranty and the software's author, the
↳holder of the economic rights, and the
15 # successive licensors have only limited liability.
16 #
17 # In this respect, the user's attention is drawn to the risks associated with loading,
↳using, modifying and/or
18 # developing or reproducing the software by the user in light of its specific status
↳of free software, that may mean
19 # that it is complicated to manipulate, and that also therefore means that it is
↳reserved for developers and
20 # experienced professionals having in-depth computer knowledge. Users are therefore
↳encouraged to load and test the
21 # software's suitability as regards their requirements in conditions enabling the
↳security of their systems and/or data
22 # to be ensured and, more generally, to use and operate it in the same conditions as
↳regards security.
23 #
24 # The fact that you are presently reading this means that you have had knowledge of
↳the CeCILL 2.1 license and that you
25 # accept its terms.
26 #*****
27 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 }}"
28 JAVA_ARGS="{{ vitam_folder_conf }}/{{ vitam_struct.vitam_component }}.conf"

```

7.2.1.2 Fichier /vitam/conf/<composant>/logback.xml

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <configuration>
3      <appender name="FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
4          <rollingPolicy class="ch.qos.logback.core.rolling.
↳TimeBasedRollingPolicy">
5              <fileNamePattern>{{ vitam_folder_log }}/accesslog-{{ vitam_
↳struct.vitam_component }}.%d{yyy-MM-dd}.log</fileNamePattern>
6              <maxHistory>{{ days_to_delete_access_local }}</maxHistory>
7              <totalSizeCap>14GB</totalSizeCap>
8          </rollingPolicy>
9          <encoder>
10             <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> (suite sur la page suivante)

```


(suite de la page précédente)

```

11         </encoder>
12     </appender>
13     <appender-ref ref="FILE" />
14 </configuration>

```

7.2.1.3 Fichier /vitam/conf/<composant>/logback-access.xml

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <configuration>
3
4     <!-- Send debug messages to System.out -->
5     <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
6         <!-- By default, encoders are assigned the type ch.qos.logback.
7         ↪ classic.encoder.PatternLayoutEncoder -->
8         <encoder>
9             <pattern>%d{ISO8601} [[%thread]] [%X{X-Request-Id}] %-5level
10            ↪ %logger - %replace(%caller{1..2}){'Caller\+1          at |\n',''} : %msg
11            ↪ %rootException%n</pattern>
12        </encoder>
13    </appender>
14    <!-- <appender name="FILE" class="ch.qos.logback.core.FileAppender" -->
15    <appender name="FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
16        <rollingPolicy class="ch.qos.logback.core.rolling.
17        ↪ SizeAndTimeBasedRollingPolicy">
18            <fileNamePattern>{{ vitam_folder_log }}/{{ vitam_struct.vitam_
19            ↪ component }}.%d{yyyy-MM-dd}.%i.log</fileNamePattern>
20            <maxFileSize>10MB</maxFileSize>
21            <maxHistory>{{ days_to_delete_logback_logfiles }}</maxHistory>
22            <totalSizeCap>5GB</totalSizeCap>
23        </rollingPolicy>
24
25        <!-- TODO : replace with rolling file appender -->
26        <!-- <file>{{ vitam_folder_log }}/{{ vitam_struct.vitam_component }}.log
27        ↪ </file>
28
29        <append>true</append> -->
30        <encoder>
31            <pattern>%d{ISO8601} [[%thread]] [%X{X-Request-Id}] %-5level
32            ↪ %logger - %replace(%caller{1..2}){'Caller\+1          at |\n',''} : %msg %rootException%n
33            ↪ </pattern>
34        </encoder>
35    </appender>
36
37    <appender name="SECURITY" class="ch.qos.logback.core.rolling.
38    ↪ RollingFileAppender">
39        <rollingPolicy class="ch.qos.logback.core.rolling.
40        ↪ SizeAndTimeBasedRollingPolicy">
41            <fileNamePattern>{{ vitam_folder_log }}/{{ vitam_struct.vitam_
42            ↪ component }}_security.%d{yyyy-MM-dd}.%i.log</fileNamePattern>
43            <maxFileSize>10MB</maxFileSize>
44            <maxHistory>{{ days_to_delete_logback_logfiles }}</maxHistory>
45            <totalSizeCap>5GB</totalSizeCap>
46        </rollingPolicy>
47        <encoder>
48            <pattern>%d{ISO8601} [[%thread]] [%X{X-Request-Id}] %-5level
49            ↪ %logger - %replace(%caller{1..2}){'Caller\+1          at |\n',''} : %msg %rootException%n

```

```

38         </pattern>
39     </encoder>
40 </appender>
41
42     {% if vitam_struct.vitam_component == 'storage' %}
43     <appender name="OFFERSYNC" class="ch.qos.logback.core.rolling.
↳RollingFileAppender">
44         <rollingPolicy class="ch.qos.logback.core.rolling.
↳SizeAndTimeBasedRollingPolicy">
45             <fileNamePattern>{{ vitam_folder_log }}/{{ vitam_struct.vitam_
↳component }}_offer_sync.%d{yyyy-MM-dd}.%i.log</fileNamePattern>
46             <maxFileSize>10MB</maxFileSize>
47             <maxHistory>30</maxHistory>
48             <totalSizeCap>5GB</totalSizeCap>
49         </rollingPolicy>
50         <encoder>
51             <pattern>%d{ISO8601} [[%thread]] [%X{X-Request-Id}] %-5level
↳%logger - %replace(%caller{1..2}){'Caller\+1      at |\n',''} : %msg %rootException%n
52             </pattern>
53         </encoder>
54     </appender>
55     {% endif %}
56
57     <appender name="SYSLOG" class="ch.qos.logback.classic.net.SyslogAppender">
58         <syslogHost>localhost</syslogHost>
59         <facility>{{ vitam_defaults.syslog_facility }}</facility>
60         <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>
61     </appender>
62     <!-- By default, the level of the root level is set to TRACE -->
63     <root level="{{ vitam_struct.log_level | default(vitam_defaults.services.log_
↳level) }}">
64         <!-- <appender-ref ref="STDOUT" /> -->
65         <appender-ref ref="FILE" />
66         <appender-ref ref="SYSLOG" />
67     </root>
68
69     <logger name="org.eclipse.jetty" level="WARN"/>
70     <logger name="fr.gouv.vitam.storage.engine.server.logbook.StorageLogbookMock" level=
↳"INFO"/>
71     <logger name="fr.gouv.vitam.metadata.core.reconstruction.ReconstructionService"
↳level="INFO"/>
72     <logger name="fr.gouv.vitam.metadata.core.graph.StoreGraphService" level="INFO"/>
73     <logger name="fr.gouv.vitam.metadata.core.graph.GraphComputeServiceImpl" level="INFO
↳"/>
74     <logger name="fr.gouv.vitam.common" level="WARN" />
75     <logger name="fr.gouv.vitam.common.performance.PerformanceLogger" level="DEBUG" />
76     <logger name="fr.gouv.vitam.common.alert.AlertServiceImpl" level="INFO">
77         <appender-ref ref="SECURITY" />
78     </logger>
79     {% if vitam_struct.vitam_component == 'storage' %}
80     <logger name="fr.gouv.vitam.storage.engine.server.offersynchronization.
↳OfferSyncServiceImpl">
81         <appender-ref ref="OFFERSYNC" />
82     </logger>
83     {% endif %}

```

(suite de la page précédente)

```

84
85     {% if vitam_struct.vitam_component == 'metadata' %}
86     <logger name="fr.gouv.vitam.metadata.core.migration" level="INFO"/>
87     {% endif %}
88
89 </configuration>

```

7.2.1.4 Fichier /vitam/conf/<composant>/jetty-config.xml

```

1 <?xml version="1.0"?>
2 <!DOCTYPE Configure PUBLIC "-//Jetty//Configure//EN" "http://www.eclipse.org/jetty/
  ↪configure_9_0.dtd">
3
4 <!-- ===== -->
5 <!-- Documentation of this file format can be found at: -->
6 <!-- http://wiki.eclipse.org/Jetty/Reference/jetty.xml_syntax -->
7 <!-- -->
8 <!-- Additional configuration files are available in $JETTY_HOME/etc -->
9 <!-- and can be mixed in. See start.ini file for the default -->
10 <!-- configuration files. -->
11 <!-- -->
12 <!-- For a description of the configuration mechanism, see the -->
13 <!-- output of: -->
14 <!-- java -jar start.jar -? -->
15 <!-- ===== -->
16
17 <!-- ===== -->
18 <!-- Configure a Jetty Server instance with an ID "Server" -->
19 <!-- Other configuration files may also configure the "Server" -->
20 <!-- ID, in which case they are adding configuration to the same -->
21 <!-- instance. If other configuration have a different ID, they -->
22 <!-- will create and configure another instance of Jetty. -->
23 <!-- Consult the javadoc of o.e.j.server.Server for all -->
24 <!-- configuration that may be set here. -->
25 <!-- ===== -->
26 <Configure id="Server" class="org.eclipse.jetty.server.Server">
27
28
29 <!-- ===== -->
30 <!-- Add shared Scheduler instance -->
31 <!-- ===== -->
32 <Call name="addBean">
33     <Arg>
34         <New class="org.eclipse.jetty.util.thread.ScheduledExecutorScheduler"/>
35     </Arg>
36 </Call>
37
38 <!-- ===== -->
39 <!-- Http Configuration. -->
40 <!-- This is a common configuration instance used by all -->
41 <!-- connectors that can carry HTTP semantics (HTTP, HTTPS, SPDY)-->
42 <!-- It configures the non wire protocol aspects of the HTTP -->
43 <!-- semantic. -->
44 <!-- -->
45 <!-- This configuration is only defined here and is used by -->

```

(suite sur la page suivante)

(suite de la page précédente)

```

46 <!-- reference from the jetty-http.xml, jetty-https.xml and -->
47 <!-- jetty-spdy.xml configuration files which instantiate the -->
48 <!-- connectors. -->
49 <!-- -->
50 <!-- Consult the javadoc of o.e.j.server.HttpConfiguration -->
51 <!-- for all configuration that may be set here. -->
52 <!-- ===== -->
53 <New id="httpConfig" class="org.eclipse.jetty.server.HttpConfiguration">
54 <Set name="secureScheme">http</Set>
55 <Set name="securePort">8443</Set>
56 <Set name="outputBufferSize">32768</Set>
57 <Set name="requestHeaderSize">8192</Set>
58 <Set name="responseHeaderSize">8192</Set>
59 <Set name="sendServerVersion">false</Set>
60 <Set name="sendDateHeader">false</Set>
61 <Set name="headerCacheSize">512</Set>
62
63 <!-- Uncomment to enable handling of X-Forwarded- style headers
64 <Call name="addCustomizer">
65 <Arg><New class="org.eclipse.jetty.server.ForwardedRequestCustomizer"/></
↪Arg>
66 </Call>
67 -->
68 </New>
69
70 <!-- ===== Original Connector ===== -
↪->
71 <!-- <Call name="addConnector">
↪ -->
72 <!-- <Arg>
↪ -->
73 <!-- <New class="org.eclipse.jetty.server.ServerConnector">
↪ -->
74 <!-- <Arg name="server"><Ref refid="Server" /></Arg>
↪ -->
75 <!-- <Arg name="factories">
↪ -->
76 <!-- <Array type="org.eclipse.jetty.server.ConnectionFactory">
↪ -->
77 <!-- <Item>
↪ -->
78 <!-- <New class="org.eclipse.jetty.server.
↪HttpConnectionFactory"> -->
79 <!-- <Arg name="config"><Ref refid="httpConfig" /></
↪Arg> -->
80 <!-- </New>
↪ -->
81 <!-- </Item>
↪ -->
82 <!-- </Array>
↪ -->
83 <!-- </Arg>
↪ -->
84 <!-- <Set name="port">{{ vitam_struct.port_service }}</Set>
↪ -->
85 <!-- <Set name="idleTimeout">
↪ -->

```

(suite sur la page suivante)

(suite de la page précédente)

```

86     <!--           <Property name="http.timeout" default="{ vitam_defaults.
↳services.port_service_timeout }"/>           -->
87     <!--           </Set>
↳
88     <!--           </New>
↳
89     <!--           </Arg>
↳
90     <!-- </Call>
↳
91
92
93
94     <!-- ===== -->
95     <!-- Set the default handler structure for the Server -->
96     <!-- A handler collection is used to pass received requests to -->
97     <!-- both the ContextHandlerCollection, which selects the next -->
98     <!-- handler by context path and virtual host, and the -->
99     <!-- DefaultHandler, which handles any requests not handled by -->
100    <!-- the context handlers. -->
101    <!-- Other handlers may be added to the "Handlers" collection, -->
102    <!-- for example the jetty-requestlog.xml file adds the -->
103    <!-- RequestLogHandler after the default handler -->
104    <!-- ===== -->
105    <Set name="handler">
106        <New id="Handlers" class="org.eclipse.jetty.server.handler.HandlerCollection">
107            <Set name="handlers">
108                <Array type="org.eclipse.jetty.server.Handler">
109                    <Item>
110                        <New id="Contexts" class="org.eclipse.jetty.server.handler.
↳ContextHandlerCollection"/>
111                    </Item>
112                    <Item>
113                        <New id="DefaultHandler" class="org.eclipse.jetty.server.
↳handler.DefaultHandler"/>
114                    </Item>
115                </Array>
116            </Set>
117        </New>
118    </Set>
119
120    <Set name="RequestLog">
121        <New id="RequestLogImpl" class="ch.qos.logback.access.jetty.RequestLogImpl
↳">
122            <Set name="fileName">{{vitam_folder_conf}}/logback-access.xml</Set>
123        </New>
124    </Set>
125    <Ref id="RequestLogImpl">
126        <Call name="start"/>
127    </Ref>
128
129    <!-- ===== -->
130    <!-- extra server options -->
131    <!-- ===== -->
132    <Set name="stopAtShutdown">true</Set>
133    <Set name="stopTimeout">5000</Set>
134    <Set name="dumpAfterStart">false</Set>

```

(suite sur la page suivante)

```

135     <Set name="dumpBeforeStop">false</Set>
136
137 {% if vitam_struct.https_enabled==true %}
138     <New id="httpsConfig" class="org.eclipse.jetty.server.HttpConfiguration">
139         <Set name="sendServerVersion">false</Set>
140         <Set name="sendDateHeader">false</Set>
141         <Call name="addCustomizer">
142             <Arg>
143                 <New class="org.eclipse.jetty.server.SecureRequestCustomizer" />
144             </Arg>
145         </Call>
146     </New>
147     <New id="sslContextFactory" class="org.eclipse.jetty.util.ssl.
↪SslContextFactory">
148         <Set name="KeyStorePath">{{ vitam_folder_conf }}/keystore_{{ vitam_struct.
↪vitam_component }}.jks</Set>
149         <Set name="KeyStorePassword">{{ password_keystore }}</Set>
150         <Set name="KeyManagerPassword">{{ password_manager_keystore }}</Set>
151         <Set name="TrustStorePath">{{ vitam_folder_conf }}/truststore_{{ vitam_struct.
↪vitam_component }}.jks</Set>
152         <Set name="TrustStorePassword">{{ password_truststore }}</Set>
153         <Set name="TrustStoreType">JKS</Set>
154         <Set name="NeedClientAuth">false</Set>
155         <Set name="WantClientAuth">>true</Set>
156         <Set name="IncludeCipherSuites">
157             <Array type="String">
158                 <Item>TLS_ECDHE.*</Item>
159                 <Item>TLS_DHE_RSA.*</Item>
160             </Array>
161         </Set>
162         <Set name="IncludeProtocols">
163             <Array type="String">
164                 <Item>TLSv1</Item>
165                 <Item>TLSv1.1</Item>
166                 <Item>TLSv1.2</Item>
167             </Array>
168         </Set>
169         <Set name="ExcludeCipherSuites">
170             <Array type="String">
171                 <Item>.*NULL.*</Item>
172                 <Item>.*RC4.*</Item>
173                 <Item>.*MD5.*</Item>
174                 <Item>.*DES.*</Item>
175                 <Item>.*DSS.</Item>
176             </Array>
177         </Set>
178         <Set name="UseCipherSuitesOrder">>true</Set>
179         <Set name="RenegotiationAllowed">>true</Set>
180     </New>
181     <New id="sslConnectionFactory" class="org.eclipse.jetty.server.
↪SslConnectionFactory">
182         <Arg name="sslContextFactory">
183             <Ref refid="sslContextFactory" />
184         </Arg>
185         <Arg name="next">http/1.1</Arg>
186     </New>
187     <New id="businessConnector" class="org.eclipse.jetty.server.ServerConnector">

```

(suite sur la page suivante)

(suite de la page précédente)

```

188     <Arg name="server">
189         <Ref refid="Server" />
190     </Arg>
191     <Arg name="factories">
192         <Array type="org.eclipse.jetty.server.ConnectionFactory">
193             <Item>
194                 <Ref refid="sslConnectionFactory" />
195             </Item>
196             <Item>
197                 <New class="org.eclipse.jetty.server.HttpConnectionFactory">
198                     <Arg name="config">
199                         <Ref refid="httpsConfig" />
200                     </Arg>
201                 </New>
202             </Item>
203         </Array>
204     </Arg>
205     <Set name="host">{{ip_service}}</Set>
206     <Set name="port">
207         <SystemProperty name="jetty.port" default="{{ vitam_struct.port_service }}"
↳"/>
208     </Set>
209     <Set name="name">business</Set>
210 </New>
211
212 {% else %}
213
214     <!-- ===== -->
215     <!-- Connector for API business -->
216     <!-- Attach all ContextHanlder except Admin -->
217     <!-- ===== -->
218
219     <New id="businessConnector" class="org.eclipse.jetty.server.ServerConnector">
220         <Arg name="server"><Ref refid="Server" /></Arg>
221         <Arg name="factories">
222             <Array type="org.eclipse.jetty.server.ConnectionFactory">
223                 <Item>
224                     <New class="org.eclipse.jetty.server.HttpConnectionFactory">
225                         <Arg name="config"><Ref refid="httpConfig" /></Arg>
226                     </New>
227                 </Item>
228             </Array>
229         </Arg>
230         <Set name="host">{{ ip_service }}</Set>
231         <Set name="port">{{ vitam_struct.port_service }}</Set>
232         <Set name="name">business</Set>
233         <Set name="idleTimeout">
234             <Property name="http.timeout" default="{{ vitam_defaults.services.port_
↳service_timeout }}" />
235         </Set>
236     </New>
237
238 {% endif %}
239
240     <!-- ===== -->
241     <!-- Connector for API Admin -->
242     <!-- Attach all ContextHanlder -->

```

(suite sur la page suivante)

```

243 <!-- ===== -->
244
245 <New id="adminConnector" class="org.eclipse.jetty.server.ServerConnector">
246   <Arg name="server"><Ref refid="Server" /></Arg>
247   <Arg name="factories">
248     <Array type="org.eclipse.jetty.server.ConnectionFactory">
249       <Item>
250         <New class="org.eclipse.jetty.server.HttpConnectionFactory">
251           <Arg name="config"><Ref refid="httpConfig" /></Arg>
252         </New>
253       </Item>
254     </Array>
255   </Arg>
256   <Set name="host">{{ ip_admin }}</Set>
257   <Set name="port">{{ vitam_struct.port_admin }}</Set>
258   <Set name="name">admin</Set>
259   <Set name="idleTimeout">
260     <Property name="http.timeout" default="{{ vitam_defaults.services.port_
↪service_timeout }}" />
261   </Set>
262 </New>
263
264
265
266
267 <Call name="setConnectors">
268   <Arg>
269     <Array type="org.eclipse.jetty.server.ServerConnector">
270       <Item>
271         <Ref refid="businessConnector" />
272       </Item>
273       <Item>
274         <Ref refid="adminConnector" />
275       </Item>
276     </Array>
277   </Arg>
278 </Call>
279
280 </Configure>

```

7.2.1.5 Fichier /vitam/conf/<composant>/logbook-client.conf

Ce fichier permet de configurer l'appel au composant logbook.

```

1 serverHost: {{ vitam.logbook.host }}
2 serverPort: {{ vitam.logbook.port_service }}

```

7.2.1.6 Fichier /vitam/conf/<composant>/server-identity.conf

```

1 identityName: {{ansible_nodename}}
2 identityRole: {{vitam_struct.vitam_component}}
3 identitySiteId: {{vitam_site_id}}

```


7.2.1.7 Fichier /vitam/conf/<composant>/antisamy-esapi.xml

```

1 <?xml version="1.0" encoding="ISO-8859-1"?>
2
3 <!--
4 W3C rules retrieved from:
5 http://www.w3.org/TR/html401/struct/global.html
6 -->
7
8 <!--
9 Slashdot allowed tags taken from "Reply" page:
10 <b> <i> <p> <br> <a> <ol> <ul> <li> <dl> <dt> <dd> <em> <strong> <tt> <blockquote>
11 ↪ <div> <code> <quote>
12 -->
13
14 <anti-samy-rules xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
15     xsi:noNamespaceSchemaLocation="antisamy.xsd">
16
17     <directives>
18         <directive name="omitXmlDeclaration" value="true"/>
19         <directive name="omitDoctypeDeclaration" value="true"/>
20         <directive name="maxInputSize" value="2000000"/>
21         <directive name="embedStyleSheets" value="false"/>
22     </directives>
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_-,:\[\]!\./\\(\)]*" />
35 ↪ <!-- force non-empty with a '+' at the end instead of '*' -->
36         <regexp name="onsiteURL" value="([\w\.\?=&;\#~]+|\#(\w)+)"/>
37         <regexp name="offsiteURL" value="(\s)*((ht|f)tp(s?)://|mailto:)[A-Za-
38 ↪ z0-9]+[~a-zA-Z0-9-_\.\#\$%&;:,\?=/\+!]*(\s)*"/>
39
40     </common-regexps>
41
42     <!--
43     Tag.name = a, b, div, body, etc.
44     Tag.action = filter: remove tags, but keep content, validate: keep content as
45 ↪ long as it passes rules, remove: remove tag and contents
46     Attribute.name = id, class, href, align, width, etc.
47     Attribute.onInvalid = what to do when the attribute is invalid, e.g., remove
48 ↪ the tag (removeTag), remove the attribute (removeAttribute), filter the tag
49 ↪ (filterTag)
50     Attribute.description = What rules in English you want to tell the users they
51 ↪ can have for this attribute. Include helpful things so they'll be able to tune
52 ↪ their HTML
53 -->

```

(suite sur la page suivante)

```

48     -->
49
50     <!--
51     Some attributes are common to all (or most) HTML tags. There aren't many that
52     qualify for this. You have to make sure there's no
53     collisions between any of these attribute names with attribute names of other
54     tags that are for different purposes.
55     -->
56
57     <common-attributes>
58
59         <attribute name="lang" description="The 'lang' attribute tells the
60         browser what language the element's attribute values and content are written in">
61             <regexp-list>
62                 <regexp value="[a-zA-Z]{2,20}" />
63             </regexp-list>
64         </attribute>
65
66         <attribute name="title" description="The 'title' attribute provides
67         text that shows up in a 'tooltip' when a user hovers their mouse over the element">
68             <regexp-list>
69                 <regexp name="htmlTitle" />
70             </regexp-list>
71         </attribute>
72
73         <attribute name="href" onInvalid="filterTag">
74             <regexp-list>
75                 <regexp name="onsiteURL" />
76                 <regexp name="offsiteURL" />
77             </regexp-list>
78         </attribute>
79
80         <attribute name="align" description="The 'align' attribute of an HTML
81         element is a direction word, like 'left', 'right' or 'center'">
82             <literal-list>
83                 <literal value="center" />
84                 <literal value="left" />
85                 <literal value="right" />
86                 <literal value="justify" />
87                 <literal value="char" />
88             </literal-list>
89         </attribute>
90
91     </common-attributes>
92
93     <!--
94     This requires normal updates as browsers continue to diverge from the W3C and
95     each other. As long as the browser wars continue
96     this is going to continue. I'm not sure war is the right word for what's
97     going on. Doesn't somebody have to win a war after
98     a while?
99     -->
100
101     <global-tag-attributes>
102         <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
123         <tag name="div" action="validate"/>
124         <tag name="i" action="validate"/>
125         <tag name="b" action="validate"/>
126         <tag name="em" action="validate"/>
127         <tag name="blockquote" action="validate"/>
128         <tag name="tt" action="validate"/>
129
130         <tag name="br" action="truncate"/>
131
132         <!-- Custom Slashdot tags, though we're trimming the idea of having a
133         ↪ possible mismatching end tag with the endtag="" attribute -->
134
135         <tag name="quote" action="validate"/>
136         <tag name="ecode" action="validate"/>
137
138         <!-- Anchor and anchor related tags -->
139
140         <tag name="a" action="validate">
141
142             <attribute name="href" onInvalid="filterTag"/>
143             <attribute name="nohref">
144                 <literal-list>
145                     <literal value="nohref"/>
146                     <literal value=""/>
147                 </literal-list>
148             </attribute>
149             <attribute name="rel">
150                 <literal-list>
151                     <literal value="nofollow"/>
152                 </literal-list>
153             </attribute>

```

(suite sur la page suivante)

```

154     </tag>
155
156     <!-- List tags -->
157
158     <tag name="ul" action="validate"/>
159     <tag name="ol" action="validate"/>
160     <tag name="li" action="validate"/>
161
162 </tag-rules>
163
164
165
166 <!-- No CSS on Slashdot posts -->
167
168 <css-rules>
169 </css-rules>
170
171
172 <html-entities>
173     <entity name="amp" cdata="&amp;"/>
174     <entity name="nbsp" cdata="&amp;#160;"/>
175
176     <entity name="iexcl" cdata="&amp;#161;"/> <!--inverted exclamation_
↪mark, U+00A1 ISOnum -->
177     <entity name="cent" cdata="&amp;#162;"/> <!--cent sign, U+00A2 ISOnum_
↪-->
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 -->

```

(suite sur la page suivante)

(suite de la page précédente)

```

194     <entity name="sup3" cdata="&#179;"/> <!--superscript three =
↳ superscript digit three= cubed, U+00B3 ISOnum -->
195     <entity name="acute" cdata="&#180;"/> <!--acute accent = spacing
↳ acute, U+00B4 ISODia -->
196     <entity name="micro" cdata="&#181;"/> <!--micro sign, U+00B5
↳ ISOnum -->
197     <entity name="para" cdata="&#182;"/> <!--pilcrow sign = paragraph
↳ sign, U+00B6 ISOnum -->
198     <entity name="middot" cdata="&#183;"/> <!--middle dot = Georgian
↳ comma = Greek middle dot, U+00B7 ISOnum -->
199     <entity name="cedil" cdata="&#184;"/> <!--cedilla = spacing
↳ cedilla, U+00B8 ISODia -->
200     <entity name="sup1" cdata="&#185;"/> <!--superscript one =
↳ superscript digit one,U+00B9 ISOnum -->
201     <entity name="ordm" cdata="&#186;"/> <!--masculine ordinal
↳ indicator, U+00BA ISOnum -->
202     <entity name="raquo" cdata="&#187;"/> <!--right-pointing double
↳ angle quotation mark = right pointing guillemet, U+00BB ISOnum -->
203     <entity name="frac14" cdata="&#188;"/> <!--vulgar fraction one
↳ quarter = fraction one quarter, U+00BC ISOnum -->
204     <entity name="frac12" cdata="&#189;"/> <!--vulgar fraction one
↳ half = fraction one half, U+00BD ISOnum -->
205     <entity name="frac34" cdata="&#190;"/> <!--vulgar fraction three
↳ quarters = fraction three quarters, U+00BE ISOnum -->
206     <entity name="iquest" cdata="&#191;"/> <!--inverted question mark
↳ = turned question mark, U+00BF ISOnum -->
207     <entity name="Agrave" cdata="&#192;"/> <!--latin capital letter A
↳ with grave = latin capital letter A grave,U+00C0 ISolat1 -->
208     <entity name="Aacute" cdata="&#193;"/> <!--latin capital letter A
↳ with acute,U+00C1 ISolat1 -->
209     <entity name="Acirc" cdata="&#194;"/> <!--latin capital letter A
↳ with circumflex,U+00C2 ISolat1 -->
210     <entity name="Atilde" cdata="&#195;"/> <!--latin capital letter A
↳ with tilde,U+00C3 ISolat1 -->
211     <entity name="Auml" cdata="&#196;"/> <!--latin capital letter A
↳ with diaeresis,U+00C4 ISolat1 -->
212     <entity name="Aring" cdata="&#197;"/> <!--latin capital letter A
↳ with ring above = latin capital letter A ring, U+00C5 ISolat1 -->
213     <entity name="AElig" cdata="&#198;"/> <!--latin capital letter AE
↳ = latin capital ligature AE, U+00C6 ISolat1 -->
214     <entity name="Ccedil" cdata="&#199;"/> <!--latin capital letter C
↳ with cedilla, U+00C7 ISolat1 -->
215     <entity name="Egrave" cdata="&#200;"/> <!--latin capital letter E
↳ with grave, U+00C8 ISolat1 -->
216     <entity name="Eacute" cdata="&#201;"/> <!--latin capital letter E
↳ with acute,U+00C9 ISolat1 -->
217     <entity name="Ecirc" cdata="&#202;"/> <!--latin capital letter E
↳ with circumflex,U+00CA ISolat1 -->
218     <entity name="Euml" cdata="&#203;"/> <!--latin capital letter E
↳ with diaeresis, U+00CB ISolat1 -->
219     <entity name="Igrave" cdata="&#204;"/> <!--latin capital letter I
↳ with grave, U+00CC ISolat1 -->
220     <entity name="Iacute" cdata="&#205;"/> <!--latin capital letter I
↳ with acute, U+00CD ISolat1 -->
221     <entity name="Icirc" cdata="&#206;"/> <!--latin capital letter I
↳ with circumflex, U+00CE ISolat1 -->
222     <entity name="Iuml" cdata="&#207;"/> <!--latin capital letter I
↳ with diaeresis, U+00CF ISolat1 -->

```

(suite sur la page suivante)

(suite de la page précédente)

```

223         <entity name="ETH" cdata="&#208;" /> <!--latin capital letter ETH,
↳ U+00D0 ISOLat1 -->
224         <entity name="Ntilde" cdata="&#209;" /> <!--latin capital letter N
↳ with tilde, U+00D1 ISOLat1 -->
225         <entity name="Ograve" cdata="&#210;" /> <!--latin capital letter O
↳ with grave, U+00D2 ISOLat1 -->
226         <entity name="Oacute" cdata="&#211;" /> <!--latin capital letter O
↳ with acute, U+00D3 ISOLat1 -->
227         <entity name="Ocirc" cdata="&#212;" /> <!--latin capital letter O
↳ with circumflex, U+00D4 ISOLat1 -->
228         <entity name="Otilde" cdata="&#213;" /> <!--latin capital letter O
↳ with tilde, U+00D5 ISOLat1 -->
229         <entity name="Ouml" cdata="&#214;" /> <!--latin capital letter O
↳ with diaeresis, U+00D6 ISOLat1 -->
230         <entity name="times" cdata="&#215;" /> <!--multiplication sign,
↳ U+00D7 ISOnum -->
231         <entity name="Oslash" cdata="&#216;" /> <!--latin capital letter O
↳ with stroke = latin capital letter O slash, U+00D8 ISOLat1 -->
232         <entity name="Ugrave" cdata="&#217;" /> <!--latin capital letter U
↳ with grave, U+00D9 ISOLat1 -->
233         <entity name="Uacute" cdata="&#218;" /> <!--latin capital letter U
↳ with acute, U+00DA ISOLat1 -->
234         <entity name="Ucirc" cdata="&#219;" /> <!--latin capital letter U
↳ with circumflex, U+00DB ISOLat1 -->
235         <entity name="Uuml" cdata="&#220;" /> <!--latin capital letter U
↳ with diaeresis, U+00DC ISOLat1 -->
236         <entity name="Yacute" cdata="&#221;" /> <!--latin capital letter Y
↳ with acute, U+00DD ISOLat1 -->
237         <entity name="THORN" cdata="&#222;" /> <!--latin capital letter
↳ THORN, U+00DE ISOLat1 -->
238         <entity name="szlig" cdata="&#223;" /> <!--latin small letter
↳ sharp s = ess-zed, U+00DF ISOLat1 -->
239         <entity name="agrave" cdata="&#224;" /> <!--latin small letter a
↳ with grave = latin small letter a grave, U+00E0 ISOLat1 -->
240         <entity name="aacute" cdata="&#225;" /> <!--latin small letter a
↳ with acute, U+00E1 ISOLat1 -->
241         <entity name="acirc" cdata="&#226;" /> <!--latin small letter a
↳ with circumflex, U+00E2 ISOLat1 -->
242         <entity name="atilde" cdata="&#227;" /> <!--latin small letter a
↳ with tilde, U+00E3 ISOLat1 -->
243         <entity name="auml" cdata="&#228;" /> <!--latin small letter a
↳ with diaeresis, U+00E4 ISOLat1 -->
244         <entity name="aring" cdata="&#229;" /> <!--latin small letter a
↳ with ring above = latin small letter a ring, U+00E5 ISOLat1 -->
245         <entity name="aelig" cdata="&#230;" /> <!--latin small letter ae =
↳ latin small ligature ae, U+00E6 ISOLat1 -->
246         <entity name="ccedil" cdata="&#231;" /> <!--latin small letter c
↳ with cedilla, U+00E7 ISOLat1 -->
247         <entity name="egrave" cdata="&#232;" /> <!--latin small letter e
↳ with grave, U+00E8 ISOLat1 -->
248         <entity name="eacute" cdata="&#233;" /> <!--latin small letter e
↳ with acute, U+00E9 ISOLat1 -->
249         <entity name="ecirc" cdata="&#234;" /> <!--latin small letter e
↳ with circumflex, U+00EA ISOLat1 -->
250         <entity name="euml" cdata="&#235;" /> <!--latin small letter e
↳ with diaeresis, U+00EB ISOLat1 -->
251         <entity name="igrave" cdata="&#236;" /> <!--latin small letter i
↳ with grave, U+00EC ISOLat1 -->

```

(suite sur la page suivante)

(suite de la page précédente)

```

252     <entity name="iacute" cdata="&#237;"/> <!--latin small letter i_
↳with acute, U+00ED ISolat1 -->
253     <entity name="icirc" cdata="&#238;"/> <!--latin small letter i_
↳with circumflex, U+00EE ISolat1 -->
254     <entity name="iuml" cdata="&#239;"/> <!--latin small letter i_
↳with diaeresis, U+00EF ISolat1 -->
255     <entity name="eth" cdata="&#240;"/> <!--latin small letter eth,_
↳U+00F0 ISolat1 -->
256     <entity name="ntilde" cdata="&#241;"/> <!--latin small letter n_
↳with tilde, U+00F1 ISolat1 -->
257     <entity name="ograve" cdata="&#242;"/> <!--latin small letter o_
↳with grave, U+00F2 ISolat1 -->
258     <entity name="oacute" cdata="&#243;"/> <!--latin small letter o_
↳with acute, U+00F3 ISolat1 -->
259     <entity name="ocirc " cdata="&#244;"/> <!--latin small letter o_
↳with circumflex, U+00F4 ISolat1 -->
260     <entity name="otilde" cdata="&#245;"/> <!--latin small letter o_
↳with tilde, U+00F5 ISolat1 -->
261     <entity name="ouml" cdata="&#246;"/> <!--latin small letter o_
↳with diaeresis, U+00F6 ISolat1 -->
262     <entity name="divide" cdata="&#247;"/> <!--division sign, U+00F7_
↳ISOnum -->
263     <entity name="oslash" cdata="&#248;"/> <!--latin small letter o_
↳with stroke, = latin small letter o slash, U+00F8 ISolat1 -->
264     <entity name="ugrave" cdata="&#249;"/> <!--latin small letter u_
↳with grave, U+00F9 ISolat1 -->
265     <entity name="uacute" cdata="&#250;"/> <!--latin small letter u_
↳with acute, U+00FA ISolat1 -->
266     <entity name="ucirc" cdata="&#251;"/> <!--latin small letter u_
↳with circumflex, U+00FB ISolat1 -->
267     <entity name="uuml" cdata="&#252;"/> <!--latin small letter u_
↳with diaeresis, U+00FC ISolat1 -->
268     <entity name="yacute" cdata="&#253;"/> <!--latin small letter y_
↳with acute, U+00FD ISolat1 -->
269     <entity name="thorn" cdata="&#254;"/> <!--latin small letter_
↳thorn, U+00FE ISolat1 -->
270     <entity name="yuml" cdata="&#255;"/> <!--latin small letter y_
↳with diaeresis, U+00FF ISolat1 -->
271
272     <entity name="fnof" cdata="&#402;"/> <!--latin small f with hook_
↳= function = florin, U+0192 ISOtech -->
273
274     <!-- Greek -->
275     <entity name="Alpha" cdata="&#913;"/> <!--greek capital letter_
↳alpha, U+0391 -->
276     <entity name="Beta" cdata="&#914;"/> <!--greek capital letter_
↳beta, U+0392 -->
277     <entity name="Gamma" cdata="&#915;"/> <!--greek capital letter_
↳gamma, U+0393 ISOgrk3 -->
278     <entity name="Delta" cdata="&#916;"/> <!--greek capital letter_
↳delta, U+0394 ISOgrk3 -->
279     <entity name="Epsilon" cdata="&#917;"/> <!--greek capital letter_
↳epsilon, U+0395 -->
280     <entity name="Zeta" cdata="&#918;"/> <!--greek capital letter_
↳zeta, U+0396 -->
281     <entity name="Eta" cdata="&#919;"/> <!--greek capital letter eta,_
↳U+0397 -->

```

(suite sur la page suivante)

(suite de la page précédente)

```

282         <entity name="Theta" cdata="&#920;"/> <!--greek capital letter_
↳theta, U+0398 ISOgrk3 -->
283         <entity name="Iota" cdata="&#921;"/> <!--greek capital letter_
↳iota, U+0399 -->
284         <entity name="Kappa" cdata="&#922;"/> <!--greek capital letter_
↳kappa, U+039A -->
285         <entity name="Lambda" cdata="&#923;"/> <!--greek capital letter_
↳lambda, U+039B ISOgrk3 -->
286         <entity name="Mu" cdata="&#924;"/> <!--greek capital letter mu,
↳U+039C -->
287         <entity name="Nu" cdata="&#925;"/> <!--greek capital letter nu,
↳U+039D -->
288         <entity name="Xi" cdata="&#926;"/> <!--greek capital letter xi,
↳U+039E ISOgrk3 -->
289         <entity name="Omicron" cdata="&#927;"/> <!--greek capital letter_
↳omicron, U+039F -->
290         <entity name="Pi" cdata="&#928;"/> <!--greek capital letter pi,
↳U+03A0 ISOgrk3 -->
291         <entity name="Rho" cdata="&#929;"/> <!--greek capital letter rho,
↳U+03A1 -->
292         <!-- there is no Sigmaf, and no U+03A2 character either -->
293         <entity name="Sigma" cdata="&#931;"/> <!--greek capital letter_
↳sigma, U+03A3 ISOgrk3 -->
294         <entity name="Tau" cdata="&#932;"/> <!--greek capital letter tau,
↳U+03A4 -->
295         <entity name="Upsilon" cdata="&#933;"/> <!--greek capital letter_
↳upsilon, U+03A5 ISOgrk3 -->
296         <entity name="Phi" cdata="&#934;"/> <!--greek capital letter phi,
↳U+03A6 ISOgrk3 -->
297         <entity name="Chi" cdata="&#935;"/> <!--greek capital letter chi,
↳U+03A7 -->
298         <entity name="Psi" cdata="&#936;"/> <!--greek capital letter psi,
↳U+03A8 ISOgrk3 -->
299         <entity name="Omega" cdata="&#937;"/> <!--greek capital letter_
↳omega, U+03A9 ISOgrk3 -->
300
301         <entity name="alpha" cdata="&#945;"/> <!--greek small letter_
↳alpha, U+03B1 ISOgrk3 -->
302         <entity name="beta" cdata="&#946;"/> <!--greek small letter beta,
↳U+03B2 ISOgrk3 -->
303         <entity name="gamma" cdata="&#947;"/> <!--greek small letter_
↳gamma, U+03B3 ISOgrk3 -->
304         <entity name="delta" cdata="&#948;"/> <!--greek small letter_
↳delta, U+03B4 ISOgrk3 -->
305         <entity name="epsilon" cdata="&#949;"/> <!--greek small letter_
↳epsilon, U+03B5 ISOgrk3 -->
306         <entity name="zeta" cdata="&#950;"/> <!--greek small letter zeta,
↳U+03B6 ISOgrk3 -->
307         <entity name="eta" cdata="&#951;"/> <!--greek small letter eta,
↳U+03B7 ISOgrk3 -->
308         <entity name="theta" cdata="&#952;"/> <!--greek small letter_
↳theta, U+03B8 ISOgrk3 -->
309         <entity name="iota" cdata="&#953;"/> <!--greek small letter iota,
↳U+03B9 ISOgrk3 -->
310         <entity name="kappa" cdata="&#954;"/> <!--greek small letter_
↳kappa, U+03BA ISOgrk3 -->
311         <entity name="lambda" cdata="&#955;"/> <!--greek small letter_
↳lambda, U+03BB ISOgrk3 -->

```

(suite sur la page suivante)

(suite de la page précédente)

```

312         <entity name="mu" cdata="&#956;"/> <!--greek small letter mu,
↳U+03BC ISOgrk3 -->
313         <entity name="nu" cdata="&#957;"/> <!--greek small letter nu,
↳U+03BD ISOgrk3 -->
314         <entity name="xi" cdata="&#958;"/> <!--greek small letter xi,
↳U+03BE ISOgrk3 -->
315         <entity name="omicron" cdata="&#959;"/> <!--greek small letter
↳omicron, U+03BF NEW -->
316         <entity name="pi" cdata="&#960;"/> <!--greek small letter pi,
↳U+03C0 ISOgrk3 -->
317         <entity name="rho" cdata="&#961;"/> <!--greek small letter rho,
↳U+03C1 ISOgrk3 -->
318         <entity name="sigmaf" cdata="&#962;"/> <!--greek small letter
↳final sigma, U+03C2 ISOgrk3 -->
319         <entity name="sigma" cdata="&#963;"/> <!--greek small letter
↳sigma, U+03C3 ISOgrk3 -->
320         <entity name="tau" cdata="&#964;"/> <!--greek small letter tau,
↳U+03C4 ISOgrk3 -->
321         <entity name="upsilon" cdata="&#965;"/> <!--greek small letter
↳upsilon, U+03C5 ISOgrk3 -->
322         <entity name="phi" cdata="&#966;"/> <!--greek small letter phi,
↳U+03C6 ISOgrk3 -->
323         <entity name="chi" cdata="&#967;"/> <!--greek small letter chi,
↳U+03C7 ISOgrk3 -->
324         <entity name="psi" cdata="&#968;"/> <!--greek small letter psi,
↳U+03C8 ISOgrk3 -->
325         <entity name="omega" cdata="&#969;"/> <!--greek small letter
↳omega, U+03C9 ISOgrk3 -->
326         <entity name="thetasym" cdata="&#977;"/> <!--greek small letter
↳theta symbol, U+03D1 NEW -->
327         <entity name="upsih" cdata="&#978;"/> <!--greek upsilon with hook
↳symbol, U+03D2 NEW -->
328         <entity name="piv" cdata="&#982;"/> <!--greek pi symbol, U+03D6
↳ISOgrk3 -->
329
330         <!-- General Punctuation -->
331         <entity name="bull" cdata="&#8226;"/> <!--bullet = black small
↳circle, U+2022 ISOpub -->
332         <!-- bullet is NOT the same as bullet operator, U+2219 -->
333         <entity name="hellip" cdata="&#8230;"/> <!--horizontal ellipsis =
↳three dot leader, U+2026 ISOpub -->
334         <entity name="prime" cdata="&#8242;"/> <!--prime = minutes = feet,
↳ U+2032 ISotech -->
335         <entity name="Prime" cdata="&#8243;"/> <!--double prime = seconds
↳= inches, U+2033 ISotech -->
336         <entity name="oline" cdata="&#8254;"/> <!--overline = spacing
↳overscore, U+203E NEW -->
337         <entity name="fracsl" cdata="&#8260;"/> <!--fraction slash, U+2044
↳NEW -->
338
339         <!-- Letterlike Symbols -->
340         <entity name="weierp" cdata="&#8472;"/> <!--script capital P =
↳power set = Weierstrass p, U+2118 ISOamso -->
341         <entity name="image" cdata="&#8465;"/> <!--blackletter capital I
↳= imaginary part, U+2111 ISOamso -->
342         <entity name="real" cdata="&#8476;"/> <!--blackletter capital R =
↳real part symbol, U+211C ISOamso -->

```

(suite sur la page suivante)

```

343         <entity name="trade" cdata="&#8482;"/> <!--trade mark sign,
↳U+2122 ISOnum -->
344         <entity name="alefsym" cdata="&#8501;"/> <!--alef symbol = first
↳transfinite cardinal, U+2135 NEW -->
345         <!-- alef symbol is NOT the same as hebrew letter alef,
346         U+05D0 although the same glyph could be used to depict both
↳characters -->
347
348         <!-- Arrows -->
349         <entity name="larr" cdata="&#8592;"/> <!--leftwards arrow, U+2190
↳ISOnum -->
350         <entity name="uarr" cdata="&#8593;"/> <!--upwards arrow, U+2191
↳ISOnum-->
351         <entity name="rarr" cdata="&#8594;"/> <!--rightwards arrow,
↳U+2192 ISOnum -->
352         <entity name="darr" cdata="&#8595;"/> <!--downwards arrow, U+2193
↳ISOnum -->
353         <entity name="harr" cdata="&#8596;"/> <!--left right arrow,
↳U+2194 ISOamsa -->
354         <entity name="crarr" cdata="&#8629;"/> <!--downwards arrow with
↳corner leftwards
355         = carriage return, U+21B5 NEW -->
356         <entity name="lArr" cdata="&#8656;"/> <!--leftwards double arrow,
↳U+21D0 ISOtech -->
357
358         <!-- ISO 10646 does not say that lArr is the same as the 'is implied
↳by' arrow
359         but also does not have any other character for that function. So ?
↳ lArr can
360         be used for 'is implied by' as ISOtech suggests -->
361
362         <entity name="uArr" cdata="&#8657;"/> <!--upwards double arrow,
↳U+21D1 ISOamsa -->
363         <entity name="rArr" cdata="&#8658;"/> <!--rightwards double arrow,
↳ U+21D2 ISOtech -->
364
365         <!-- ISO 10646 does not say this is the 'implies' character but does
↳not have
366         another character with this function so ?
367         rArr can be used for 'implies' as ISOtech suggests -->
368
369         <entity name="dArr" cdata="&#8659;"/> <!--downwards double arrow,
↳U+21D3 ISOamsa -->
370         <entity name="hArr" cdata="&#8660;"/> <!--left right double arrow,
↳ U+21D4 ISOamsa -->
371
372         <!-- Mathematical Operators -->
373         <entity name="forall" cdata="&#8704;"/> <!--for all, U+2200
↳ISOtech -->
374         <entity name="part" cdata="&#8706;"/> <!--partial differential,
↳U+2202 ISOtech -->
375         <entity name="exist" cdata="&#8707;"/> <!--there exists, U+2203
↳ISOtech -->
376         <entity name="empty" cdata="&#8709;"/> <!--empty set = null set =
↳diameter,U+2205 ISOamso -->
377         <entity name="nabla" cdata="&#8711;"/> <!--nabla = backward
↳difference, U+2207 ISOtech -->

```

(suite sur la page suivante)

(suite de la page précédente)

```

378         <entity name="isin" cdata="&#8712;"/> <!--element of, U+2208,
↳ ISOtech -->
379         <entity name="notin" cdata="&#8713;"/> <!--not an element of,
↳ U+2209 ISOtech -->
380         <entity name="ni" cdata="&#8715;"/> <!--contains as member,
↳ U+220B ISOtech -->
381
382         <!-- should there be a more memorable name than 'ni'? -->
383         <entity name="prod" cdata="&#8719;"/> <!--n-ary product = product,
↳ sign, U+220F ISOamsb -->
384
385         <!-- prod is NOT the same character as U+03A0 'greek capital letter pi
↳ ' though
386             the same glyph might be used for both -->
387
388         <entity name="sum" cdata="&#8721;"/> <!--n-ary sumation, U+2211,
↳ ISOamsb -->
389
390         <!-- sum is NOT the same character as U+03A3 'greek capital letter,
↳ sigma'
391             though the same glyph might be used for both -->
392
393         <entity name="minus" cdata="&#8722;"/> <!--minus sign, U+2212,
↳ ISOtech -->
394         <entity name="lowast" cdata="&#8727;"/> <!--asterisk operator,
↳ U+2217 ISOtech -->
395         <entity name="radic" cdata="&#8730;"/> <!--square root = radical,
↳ sign, U+221A ISOtech -->
396         <entity name="prop" cdata="&#8733;"/> <!--proportional to, U+221D,
↳ ISOtech -->
397         <entity name="infin" cdata="&#8734;"/> <!--infinity, U+221E,
↳ ISOtech -->
398         <entity name="ang" cdata="&#8736;"/> <!--angle, U+2220 ISOamso -->
399         <entity name="and" cdata="&#8743;"/> <!--logical and = wedge,
↳ U+2227 ISOtech -->
400         <entity name="or" cdata="&#8744;"/> <!--logical or = vee, U+2228,
↳ ISOtech -->
401         <entity name="cap" cdata="&#8745;"/> <!--intersection = cap,
↳ U+2229 ISOtech -->
402         <entity name="cup" cdata="&#8746;"/> <!--union = cup, U+222A,
↳ ISOtech -->
403         <entity name="int" cdata="&#8747;"/> <!--integral, U+222B ISOtech,
↳ -->
404         <entity name="there4" cdata="&#8756;"/> <!--therefore, U+2234,
↳ ISOtech -->
405         <entity name="sim" cdata="&#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="&#8773;"/> <!--approximately equal to,
↳ U+2245 ISOtech -->
411         <entity name="asymp" cdata="&#8776;"/> <!--almost equal to =,
↳ asymptotic to, U+2248 ISOamsr -->
412         <entity name="ne" cdata="&#8800;"/> <!--not equal to, U+2260,
↳ ISOtech -->

```

(suite sur la page suivante)

```

413     <entity name="equiv" cdata="&#8801;"/> <!--identical to, U+2261_
↳ISOtech -->
414     <entity name="le" cdata="&#8804;"/> <!--less-than or equal to,
↳U+2264 ISOtech -->
415     <entity name="ge" cdata="&#8805;"/> <!--greater-than or equal to,
↳U+2265 ISOtech -->
416     <entity name="sub" cdata="&#8834;"/> <!--subset of, U+2282_
↳ISOtech -->
417     <entity name="sup" cdata="&#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="&#8836;"/> <!--not a subset of, U+2284_
↳ISOamsn -->
424     <entity name="sube" cdata="&#8838;"/> <!--subset of or equal to,
↳U+2286 ISOtech -->
425     <entity name="supe" cdata="&#8839;"/> <!--superset of or equal to,
↳ U+2287 ISOtech -->
426     <entity name="oplus" cdata="&#8853;"/> <!--circled plus = direct_
↳sum, U+2295 ISOamsb -->
427     <entity name="otimes" cdata="&#8855;"/> <!--circled times =
↳vector product, U+2297 ISOamsb -->
428     <entity name="perp" cdata="&#8869;"/> <!--up tack = orthogonal to_
↳= perpendicular, U+22A5 ISOtech -->
429     <entity name="sdot" cdata="&#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="&#8968;"/> <!--left ceiling = apl_
↳upstile, U+2308 ISOamsc -->
434     <entity name="rceil" cdata="&#8969;"/> <!--right ceiling, U+2309_
↳ISOamsc -->
435     <entity name="lfloor" cdata="&#8970;"/> <!--left floor = apl_
↳downstile, U+230A ISOamsc -->
436     <entity name="rfloor" cdata="&#8971;"/> <!--right floor, U+230B_
↳ISOamsc -->
437     <entity name="lang" cdata="&#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="&#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="&#9674;"/> <!--lozenge, U+25CA ISOpub --
↳>
445
446     <!-- Miscellaneous Symbols -->
447     <entity name="spades" cdata="&#9824;"/> <!--black spade suit,
↳U+2660 ISOpub -->

```

(suite de la page précédente)

```

448         <!-- black here seems to mean filled as opposed to hollow -->
449         <entity name="clubs" cdata="&#9827;" /> <!--black club suit =
↳shamrock, U+2663 ISOpub -->
450         <entity name="hearts" cdata="&#9829;" /> <!--black heart suit =
↳valentine, U+2665 ISOpub -->
451         <entity name="diams" cdata="&#9830;" /> <!--black diamond suit,
↳U+2666 ISOpub -->
452
453         <entity name="quot" cdata="&#34;" /> <!--quotation mark = API
↳quote, U+0022 ISOnum -->
454         <!-- Latin Extended-A -->
455         <entity name="OElig" cdata="&#338;" /> <!--latin capital ligature
↳OE, U+0152 ISOLat2 -->
456         <entity name="oelig" cdata="&#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="&#352;" /> <!--latin capital letter
↳S with caron, U+0160 ISOLat2 -->
459         <entity name="scaron" cdata="&#353;" /> <!--latin small letter s
↳with caron, U+0161 ISOLat2 -->
460         <entity name="Yuml" cdata="&#376;" /> <!--latin capital letter Y
↳with diaeresis, U+0178 ISOLat2 -->
461
462         <!-- Spacing Modifier Letters -->
463         <entity name="circ" cdata="&#710;" /> <!--modifier letter
↳circumflex accent, U+02C6 ISOpub -->
464         <entity name="tilde" cdata="&#732;" /> <!--small tilde, U+02DC
↳ISODia -->
465
466         <!-- General Punctuation -->
467         <entity name="ensp" cdata="&#8194;" /> <!--en space, U+2002 ISOpub
↳-->
468         <entity name="emsp" cdata="&#8195;" /> <!--em space, U+2003 ISOpub
↳-->
469         <entity name="thinsp" cdata="&#8201;" /> <!--thin space, U+2009
↳ISOpub -->
470         <entity name="zwnj" cdata="&#8204;" /> <!--zero width non-joiner,
↳U+200C NEW RFC 2070 -->
471         <entity name="zwj" cdata="&#8205;" /> <!--zero width joiner,
↳U+200D NEW RFC 2070 -->
472         <entity name="lrm" cdata="&#8206;" /> <!--left-to-right mark,
↳U+200E NEW RFC 2070 -->
473         <entity name="rlm" cdata="&#8207;" /> <!--right-to-left mark,
↳U+200F NEW RFC 2070 -->
474         <entity name="ndash" cdata="&#8211;" /> <!--en dash, U+2013 ISOpub
↳-->
475         <entity name="mdash" cdata="&#8212;" /> <!--em dash, U+2014 ISOpub
↳-->
476         <entity name="lsquo" cdata="&#8216;" /> <!--left single quotation
↳mark, U+2018 ISOnum -->
477         <entity name="rsquo" cdata="&#8217;" /> <!--right single quotation
↳mark, U+2019 ISOnum -->
478         <entity name="sbquo" cdata="&#8218;" /> <!--single low-9 quotation
↳mark, U+201A NEW -->
479         <entity name="ldquo" cdata="&#8220;" /> <!--left double quotation
↳mark, U+201C ISOnum -->

```

(suite sur la page suivante)

(suite de la page précédente)

```

480         <entity name="rdquo" cdata="&#8221;" /> <!--right double quotation_
↳mark, U+201D ISOnum -->
481         <entity name="bdquo" cdata="&#8222;" /> <!--double low-9 quotation_
↳mark, U+201E NEW -->
482         <entity name="dagger" cdata="&#8224;" /> <!--dagger, U+2020 ISOpub_
↳-->
483         <entity name="Dagger" cdata="&#8225;" /> <!--double dagger, U+2021_
↳ISOpub -->
484         <entity name="permil" cdata="&#8240;" /> <!--per mille sign, _
↳U+2030 ISotech -->
485         <entity name="lsaquo" cdata="&#8249;" /> <!--single left-pointing_
↳angle quotation mark, U+2039 ISO proposed -->
486         <!-- lsaquo is proposed but not yet ISO standardized -->
487         <entity name="rsaquo" cdata="&#8250;" /> <!--single right-pointing_
↳angle quotation mark, U+203A ISO proposed -->
488         <!-- rsaquo is proposed but not yet ISO standardized -->
489         <entity name="euro" cdata="&#8364;" /> <!--euro sign, U+20AC NEW -
↳->
490     </html-entities>
491
492 </anti-samy-rules>

```

7.2.1.8 Fichier /vitam/conf/<composant>/vitam.conf

```

1 secret : {{plateforme_secret}}
2 filterActivation : {{ vitam_struct.secret_platform }}
3 {% if vitam_struct.vitam_component == vitam.processing.vitam_component %}
4 distributeurBatchSize: 800
5 workerBulkSize: 16
6 {% endif %}
7 {% if vitam_struct.vitam_component == vitam.metadata.vitam_component %}
8 storeGraphElementsPerFile: 10000
9 storeGraphOverlapDelay: 300
10 expireCacheEntriesDelay: 300
11 deleteIncompleteReconstructedUnitDelay: 2592000
12 migrationBulkSize: 10000
13 {% endif %}
14 distributionThreshold : 100000
15 eliminationAnalysisThreshold : 100000
16 eliminationActionThreshold : 10000
17 intervalDelayCheckIdle : 5000
18 maxDelayUnusedConnection : 5000
19 delayValidationAfterInactivity : 2500
20 tenants: [ "{{ vitam_tenant_ids | join(' ', ' ') }}" ]
21 adminTenant : {{vitam_tenant_admin}}
22 forceChunkModeInputStream : {{vitam_defaults.vitam_force_chunk_mode}}
23
24 {% if vitam_struct.vitam_component == vitam.worker.vitam_component %}
25 reclassificationMaxBulkThreshold: 1000
26 reclassificationMaxUnitsThreshold: 10000
27 reclassificationMaxGuildListSizeInLogbookOperation: 1000
28 {% endif %}
29
30 keywordMaxLength: 32766
31 textMaxLength: 32766

```

(suite sur la page suivante)

(suite de la page précédente)

```

32
33 classificationLevel :
34   allowList : [{% for classification in classificationList %}{% classification %}{%
↳if not loop.last %},{% endif %}{% endfor %}]
35   authorizeNotDefined: {{classificationLevelOptional}}
36

```

Ce fichier permet de définir le secret de plate-forme.

7.2.1.9 Fichier /vitam/conf/<composant>/vitam.metrics.conf

```

1 # Fichier de configuration des métriques
2 #
3 # Les différents clés disponibles pour ce fichier de configuration sont les
↳suitantes :
4 #
5 # metricsJersey: true / false           Active ou non les métriques Jersey
6 # metricsJVM: true / false             Active ou non les métriques JVM
7 #
8 # metricReporter: ELASTICSEARCH | LOGBACK | NONE           défini le
↳type de reporter
9 # metricReporterInterval: int > 0           défini l
↳'interval entre chaque reporting
10 # metricReporterIntervalUnit: TimeUnit (ex: SECONDS, MINUTES...)   défini le
↳type d'interval
11 #
12 # Si le reporter est de type LOGBACK, la clé suivante est configurable:
13 # metricLogLevel: DEBUG | INFO | WARN | ERROR ...           défini le
↳niveau de log Logback
14 #
15 # Si le reporter est de type ELASTICSEARCH, la clé suivante est obligatoire :
16 #
17 # (un tableau avec les différentes adresses des bases ElasticSearch)
18 # metricReporterHosts:
19 #     - 127.0.0.1:9201
20 #     - 0.0.0.0:80
21 #     - 8.8.8.8:22
22
23 {% if (groups['hosts-logstash'] | length) > 0 %}
24 metricsJersey: true
25 metricsJVM: true
26
27 metricReporter: ELASTICSEARCH
28 metricReporterHosts:
29 {% for host in groups['hosts-elasticsearch-log'] %}
30   - "{{ hostvars[host]['ip_admin'] }}:{{ elasticsearch.log.port_http }}"
31 {% endfor %}
32 metricLogLevel: DEBUG
33 metricReporterInterval: 3
34 metricReporterIntervalUnit: MINUTES
35 {% endif %}

```

7.2.1.10 Fichier `/vitam/conf/<composant>/java.security`

```
1 # Use Bouncy Castle Provider when it is available
2 security.provider.9=org.bouncycastle.jce.provider.BouncyCastleProvider
3
4 # Override the default list of Centos 7 that disable Elliptic Curved Based Algorithms
5 jdk.tls.disabledAlgorithms="SSLv3, RC4, MD5withRSA, DH keySize < 768,RSA keySize <
  ↪2048"
```

7.2.2 access external

7.2.2.1 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

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*.

Les fichiers de configuration sont définis sous `/vitam/conf/access-external`.

7.2.2.2.1 Fichier `access-external.conf`

```
authentication: true
jettyConfig: jetty-config.xml
tenantFilter : true
```

7.2.2.2.2 Fichier `access-internal-client.conf`

```
serverHost: {{vitam.accessinternal.host}}
serverPort: {{vitam.accessinternal.port_service}}
```

7.2.2.2.3 Fichier `functional-administration-client.conf`

```
serverHost: {{vitam.functional_administration.host}}
serverPort: {{vitam.functional_administration.port_service}}
```


7.2.2.3 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

7.2.3 access-internal

7.2.3.1 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'une *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.

7.2.3.2 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/access`.

7.2.3.2.1 Fichier `access.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
```

7.2.3.2 Fichier storage-client.conf

Ce fichier permet de définir l'accès au storage-engine.

```
serverHost: {{ vitam.storageengine.host }}
serverPort: {{ vitam.storageengine.port_service }}
```

7.2.3.3 Fichier metadata-client.conf

Ce fichier permet de définir l'accès au storage-engine.

```
serverHost: {{ vitam.metadata.host }}
serverPort: {{ vitam.metadata.port_service }}
```

7.2.3.3 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

7.2.4 Cerebro

7.2.4.1 Présentation

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

7.2.4.2 Configuration / fichiers utiles

7.2.4.2.1 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.
# 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 %}
}

```

(suite sur la page suivante)

```

}

# A list of known hosts
hosts = [
  # {
  #   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"
  #   }
  # }
]

```

7.2.4.3 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

7.2.5 common-plugin

7.2.5.1 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 .

7.2.5.2 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`.

7.2.5.2.1 Classe Item Status

Ce classe permet de retourner le statut d'un Item.

7.2.5.2.2 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`.

7.2.5.2.3 Classe ParameterHelper

Ce classe permet de faire un check sur les paramètres et avoir le tenant parameter de session vitam .

7.2.5.2.4 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`.

7.2.5.2.5 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`.

7.2.5.2.6 Classe IOPParameter

Ce class permet de définir les paramètres Input et Output pour une action et une step .

7.2.5.2.7 Classe `ProcessingUri`

Ce classe permet de formater le processing URI .

7.2.5.2.8 Classe `UriPrefix`

C'est le Handler IO

Les classes des paramètres common sont définis sous `/vitam/processing/common/parameter`.

7.2.5.2.9 Classe `AbstractWorkerParameters`

C'est une implémentation abstraite de tous les paramètres de workers .

7.2.5.2.10 Classe `DefaultWorkerParameters`

Ce classe permet de définir les paramètres par défaut d'un worker.

7.2.5.2.11 Classe `WorkerParameterName`

Ce classe inclut une énumération avec tous les noms des paramètres d'un worker .

7.2.5.2.12 Classe `WorkerParameters`

Ce classe permet de définir les paramètres de worker.

7.2.5.2.13 Classe `WorkerParametersDeserializer`

Ce classe permet de définir les paramètres d'un worker deserializer.

7.2.5.2.14 Classe `WorkerParametersFactory`

Ce classe permet de définir les paramètres d'un worker Factory.

7.2.5.2.15 Classe `WorkerParametersSerializer`

Ce classe permet de définir les paramètres de Worker Serializer.

Les classes de model sont définis sous `/vitam/worker/common`.

7.2.5.2.16 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`.

7.2.5.2.17 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.

7.2.5.2.18 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

7.2.6 consul

7.2.6.1 Présentation

Consul est un DNS applicatif.

7.2.6.1.1 Cas serveur

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

7.2.6.1.2 Cas agent

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

7.2.6.2 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*.

7.2.6.2.1 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/service-<composant>.json` et est basé sur ce squelette :

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

```

1  {
2  "service": {
3  {% if vitam_struct.vitam_component == vitam.storageofferdefault.vitam_component %}
4      "name": "{{ offer_conf }}",
5  {% else %}
6      "name": "{{ vitam_struct.vitam_component }}",
7  {% endif %}
8      "address": "{{ ip_service }}",
9  {% if ip_wan is defined %}
10     "advertise_addr_wan": "{{ ip_wan }}",
11 {% endif %}
12 {% if vitam_struct.https_enabled==true %}
13     "port": {{ vitam_struct.port_service }},
14 {% else %}
15     "port": {{ vitam_struct.port_service }},
16 {% endif %}
17     "enable_tag_override": false,
18     "tags": ["vitam", "{{ vitam_struct.vitam_component }}"],
19     "checks": [
20         {
21             "name": "{{ vitam_struct.vitam_component }}: business service check",
22 {% if vitam_struct.https_enabled==true %}
23             "notes": "HTTPS port opened",
24             "tcp": "{{ip_service}}:{{ vitam_struct.port_service }}",
25 {% else %}
26             "notes": "HTTP port opened",
27             "tcp": "{{ip_service}}:{{ vitam_struct.port_service }}",
28 {% endif %}
29             "interval": "1s"
30         },
31         {
32             "name": "{{ vitam_struct.vitam_component }} : admin service check",
33             "notes": "Status admin : /admin/v1/status",
34             "http": "http://{{ip_admin}}:{{ vitam_struct.port_admin }}/admin/v1/status",
35             "interval": "1s"
36         }
37 {% if (vitam_struct.vitam_component == vitam.worker.vitam_component) or (vitam_struct.
↪vitam_component == vitam.ingestexternal.vitam_component) %}
38     ,
39     {
40         "name": "Siegfried check",
41         "notes": "Is siegfried running ?",
42         "tcp": "localhost:{{ siegfried.port }}",
43         "interval": "1s"
44     }
45 {% endif %}
46 {% if vitam_struct.antivirus is defined %}
47     ,
48     {

```

(suite sur la page suivante)

(suite de la page précédente)

```

49     "name": "Antivirus check",
50     "notes": "Is {{ vitam_struct.antivirus }} running ?",
51     "args": [{"{{ vitam_folder_conf }}/scan-{{ vitam_struct.antivirus}}.sh", "{{
↪vitam_folder_conf }}/scan-{{ vitam_struct.antivirus}}.sh"},
52     "interval": "30s",
53     "timeout": "5s"
54   }
55 {% endif %}
56   ]
57 }
58 }

```

7.2.6.3 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`

Avvertissement : en cas de redémarrage du cluster serveur consul, il faut procéder à un arrêt/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 visionner 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

7.2.7 Kibana interceptor

7.2.7.1 Présentation

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

7.2.7.2 Configuration / fichiers utiles

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

7.2.7.2.1 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"]
```

7.2.7.3 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.

7.2.8 elasticsearch chaîne de log

7.2.8.1 Présentation

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.

7.2.8.2 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`.

7.2.8.2.1 Fichier `log4j2.properties`

```
status = error

# log action execution errors for easier debugging
logger.action.name = org.elasticsearch.action
logger.action.level = debug

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.}] %marker%. -10000m%n
appender.rolling.filePattern = ${sys:es.logs.base_path}${sys:file.separator}${sys:es.
↳logs.cluster_name}-${d{yyyy-MM-dd)}.log
appender.rolling.policies.type = Policies
appender.rolling.policies.time.type = TimeBasedTriggeringPolicy
```

(suite sur la page suivante)

```

appender.rolling.policies.time.interval = 1
appender.rolling.policies.time.modulate = true

rootLogger.level = info
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.}] %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.size.type = SizeBasedTriggeringPolicy
appender.deprecation_rolling.policies.size.size = 1GB
appender.deprecation_rolling.strategy.type = DefaultRolloverStrategy
appender.deprecation_rolling.strategy.max = 4

logger.deprecation.name = org.elasticsearch.deprecation
logger.deprecation.level = warn
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]
↳%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

logger.index_search_slowlog_rolling.name = index.search.slowlog
logger.index_search_slowlog_rolling.level = trace
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]
↳%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

```

(suite sur la page suivante)

(suite de la page précédente)

```

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

logger.index_indexing_slowlog.name = index.indexing_slowlog.index
logger.index_indexing_slowlog.level = trace
logger.index_indexing_slowlog.appenderRef.index_indexing_slowlog_rolling.ref = index_
↳indexing_slowlog_rolling
logger.index_indexing_slowlog.additivity = false

```

7.2.8.2.2 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 }}
#
# ----- 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:
#

```

(suite sur la page suivante)

```

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 == "localhost" %}
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 -----

```

(suite de la page précédente)

```

#
# Block initial recovery after a full cluster restart until N nodes are started:
#
# 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

```

7.2.8.2.3 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.yml

# 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=

#####
# 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

```

(suite sur la page suivante)

```

# 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

#####
# 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
#####

# 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

```


(suite de la page précédente)

```
#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
```

7.2.8.2.4 Fichier /usr/lib/tmpfiles.d/elasticsearch-data.conf

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

7.2.8.3 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 ¹⁸

- actions récurrentes
- cas des batches

N/A

<https://www.elastic.co/guide/en/elasticsearch/reference/2.4/indices-open-close.html>

7.2.9 elasticsearch Vitam

7.2.9.1 Présentation

Elasticsearch-log 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.

7.2.9.2 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`.

7.2.9.2.1 Fichier `log4j2.properties`

```
status = error

# log action execution errors for easier debugging
logger.action.name = org.elasticsearch.action
logger.action.level = debug

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.}] %marker%-1000m%n
appender.rolling.filePattern = ${sys:es.logs.base_path}${sys:file.separator}${sys:es.
↳logs.cluster_name}-${d{yyyy-MM-dd}}.log
appender.rolling.policies.type = Policies
appender.rolling.policies.time.type = TimeBasedTriggeringPolicy
appender.rolling.policies.time.interval = 1
appender.rolling.policies.time.modulate = true
```

(suite sur la page suivante)

(suite de la page précédente)

```

rootLogger.level = info
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.}] %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.size.type = SizeBasedTriggeringPolicy
appender.deprecation_rolling.policies.size.size = 1GB
appender.deprecation_rolling.strategy.type = DefaultRolloverStrategy
appender.deprecation_rolling.strategy.max = 4

logger.deprecation.name = org.elasticsearch.deprecation
logger.deprecation.level = warn
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]
↳%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

logger.index_search_slowlog_rolling.name = index.search.slowlog
logger.index_search_slowlog_rolling.level = trace
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]
↳%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

```

(suite sur la page suivante)

(suite de la page précédente)

```

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

logger.index_indexing_slowlog.name = index.indexing_slowlog.index
logger.index_indexing_slowlog.level = trace
logger.index_indexing_slowlog.appenderRef.index_indexing_slowlog_rolling.ref = index_
↳indexing_slowlog_rolling
logger.index_indexing_slowlog.additivity = false

```

7.2.9.2.2 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 }}
#
# ----- 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 }}

```

(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
network.host: 0.0.0.0 # KWA : For now, keep 0.0.0.0 as vitam component use the_
↳service interface, but cerebro uses the admin interface
{% if inventory_hostname == "localhost" %}
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>
#
# ----- 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_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: {{ ((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 -----

```

(suite sur la page suivante)

```

#
# 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 performamnce 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:
  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 }}

```

(suite sur la page suivante)

(suite de la page précédente)

```
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)
```

7.2.9.2.3 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.yml

# 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=

#####
# 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"
```

(suite sur la page suivante)

```

# 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

#####
# 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
#####

# 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

```


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

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

7.2.9.3 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

7.2.10 functional administration

7.2.10.1 Présentation

Rôle :

- Gérer les référentiels métier de la plate-forme

Fonctions :

- Gestion du référentiel des formats (PRONOM)
- Gestion des règles de gestion des archives

7.2.10.2 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`.

7.2.10.2.1 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:
  0:
    - INGEST_CONTRACT
    - ACCESS_CONTRACT
  1:
    - INGEST_CONTRACT
    - ACCESS_CONTRACT
    - PROFILE
    - SECURITY_PROFILE
    - CONTEXT

listMinimumRuleDuration:
  2:
    AppraisalRule : 1 year
```

7.2.10.2.2 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- pour 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
- `CONTEXT` : contextes applicatifs
- `ARCHIVEUNITPROFILE` : 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 débiter 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

7.2.10.3 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

7.2.11 ihm-demo

7.2.11.1 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
- Formulaire dynamiques
- Suivi des opérations
- Gestion des référentiels

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*.

Les fichiers de configuration sont définis sous `/vitam/conf/ihm-demo`.

7.2.11.2.1 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
```

7.2.11.2.2 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
```

tenants : liste des tenants disponibles sur l'ihm-demo.

7.2.11.2.3 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
```

7.2.11.2.4 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
contextFactory.url = {{ ldap_authentication.ldap_protocol }}://{{ ldap_
↳ authentication.ldap_server }}:{{ ldap_authentication.ldap_port }}
{% if ldap_authentication.ldap_login is defined and ldap_authentication.ldap_pwd_
↳ is defined %}
{% if ldap_authentication.ldap_login != "" and ldap_authentication.ldap_pwd != ""
↳ %}
contextFactory.systemUsername = {{ ldap_authentication.ldap_login }}
contextFactory.systemPassword = {{ ldap_authentication.ldap_pwd }}
{% endif %}
{% endif %}

ldapRealm = fr.gouv.vitam.common.auth.core.realm.LdapRealm
ldapRealm.ldapContextFactory = $contextFactory
ldapRealm.searchBase = "{{ ldap_authentication.ldap_base }}"
ldapRealm.groupRequestFilter = {{ ldap_authentication.ldap_group_request }}
ldapRealm.userDnTemplate = {{ ldap_authentication.ldap_userDn_Template }}
```

(suite sur la page suivante)

(suite de la page précédente)

```

ldapRealm.groupRolesMap = "{{ ldap_authentication.ldap_admin_group }}" : "admin", "{{
↳ ldap_authentication.ldap_user_group }}" : "user", "{{ ldap_authentication.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]
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

```

(suite sur la page suivante)

```
# 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
```

7.2.11.3 Configuration de apache shiro

7.2.11.4 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
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
```

7.2.11.5 Décryptage de shiro.ini

[main] Contient la déclaration des options et mappings dans l'authentification ldap :

- contextFactory.url : url du serveur ldap
- contextFactory.systemUsername : identifiant de l'util

- `contextFactory.systemPassword` : mot de passe
- `realm.searchBase` : la 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 groupes qui ont la même rôle admin avec ce syntaxe :

```
"groupeA" : "admin", "groupeB" : "admin", "groupeC" : "admin"
```

7.2.11.6 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

7.2.12 ihm-recette

7.2.12.1 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 !

7.2.12.2 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`.

7.2.12.2.1 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
```

7.2.12.2.2 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
secureMode:
{% 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 %}

```

7.2.12.2.3 Fichier ihm-recette-client.conf

```

serverHost: {{ vitam_struct.host }}
serverPort: {{ vitam_struct.port_service }}

```

7.2.12.2.4 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
```

7.2.12.2.5 Fichier functional-administration-client.conf

```
serverHost: {{ vitam.functional_administration.host }}
serverPort: {{ vitam.functional_administration.port_service }}
```

7.2.12.2.6 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
```

(suite sur la page suivante)

(suite de la page précédente)

```

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 %}
[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 %}

```

7.2.12.2.7 Fichier storage-client.conf

```

serverHost: {{ vitam.storageengine.host }}
serverPort: {{ vitam.storageengine.port_service }}

```

7.2.12.2.8 Fichier storage-offer.conf

```

strategy_name=[{% for item in vitam_strategy %}"{{ item.name }}.service.{{ consul_
↳domain }}"{% if not loop.last %},{% endif %}{% endfor %}]

```

7.2.12.2.9 Fichier `tnr.conf`

```
urlWorkspace: {{vitam.workspace | client_url}}
tenantsTest: [ "0" ]
vitamSecret: {{plateforme_secret}}
```

7.2.12.3 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

7.2.13 ingest-external

7.2.13.1 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

7.2.13.2 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`.

7.2.13.2.1 Fichier `ingest-external.conf`

```
path: {{vitam_folder_data}}
jettyConfig: jetty-config.xml
authentication: true
tenantFilter : true
antiVirusScriptName: scan-{{ vitam_struct.antivirus }}.sh
timeoutScanDelay: 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.

7.2.13.2.2 Fichier `ingest-internal-client.conf`

```
serverHost: {{ vitam.ingestinternal.host }}
serverPort: {{ vitam.ingestinternal.port_service }}
```

7.2.13.3 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

7.2.14 ingest-internal

7.2.14.1 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

7.2.14.2 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`.

7.2.14.2.1 Fichier ingest-internal.conf

```
workspaceUrl: {{vitam.workspace | client_url}}
processingUrl: {{vitam.processing | client_url}}
jettyConfig: jetty-config.xml
```

7.2.14.3 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

7.2.15 log server

7.2.15.1 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.

7.2.15.2 Configuration / fichiers utiles

L'ansibleerie se charge du paramétrage de ces composants.

7.2.15.3 Opérations

- Démarrage du service

En tant qu'utilisateur root :

Pré-requis : le cluster elasticsearch associé est déjà démarré.

```
systemctl start logstash
systemctl start kibana
```

- Arrêt du service

En tant qu'utilisateur root :

```
systemctl stop kibana
systemctl stop logstash
```

Post-requis : le cluster elasticsearch associé est arrêté.

- 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>/app/kibana

- 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

7.2.16 logbook

7.2.16.1 Présentation

Rôle :

- Gérer les journaux métiers à fort besoin d'intégrité et potentiellement à valeur probante : journal du cycle de vie, journal métier (SAE/opérations + écritures)

Fonctions :

- Appel uniquement à partir de l'application

7.2.16.2 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/logbook`.

7.2.16.2.1 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"]

```

(suite sur la page suivante)

```
# Configuration des alertes de securite
alertEvents:
- 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 }}
```

7.2.16.3 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

7.2.17 metadata

7.2.17.1 Présentation

Rôle :

- Stocker de manière requêtable et rapide les métadonnées des objets (également stockées mais de manière pérenne dans l'offre de stockage)

Fonctions :

- Fournit une API agréant une technologie de base de données et un moteur d'indexation
- Fournit un cache des requêtes pour optimisation

7.2.17.2 Configuration / fichiers utiles

7.2.17.2.1 Fichier metadata.conf

```
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 %}
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 }}
```

7.2.17.3 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

7.2.18 mongoC

7.2.18.1 Présentation

Replicaset mongoDB servant à stocker la configuration MongoDB (clés de sharding, shards, ...) lors de l'utilisation de MondoDB en mode sharding.

7.2.18.2 Configuration / fichiers utiles

Pour le moment, aucun fichier à paramétrer.

7.2.18.3 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 ?

7.2.19 mongoD

7.2.19.1 Présentation

Replicaset MongoDB stockant les données métier de Vitam.

7.2.19.2 Configuration / fichiers utiles

Pour le moment, aucun fichier à paramétrer.

7.2.19.3 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 ?

7.2.20 mongoS

7.2.20.1 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.

7.2.20.2 Configuration / fichiers utiles

Pour le moment, aucun fichier à paramétrer.

7.2.20.3 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 ?

7.2.21 processing

7.2.21.1 Présentation

Rôle :

- Exécution massive de processus métiers complexes
- Utilisé notamment lors du versement et de la préservation

Fonctions :

- Découpage en micro tâches de processus métier (en fonction d'un référentiel)
- Supervision de l'état d'exécution de chaque « job »
- Reprise sur incident
- Traçabilité de l'ensemble des actions effectuées

7.2.21.2 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/processing`.

7.2.21.2.1 Fichier `processing.conf`

```
urlMetadata: {{ vitam_struct | client_url }}
urlWorkspace: {{ vitam_workspace | client_url }}
jettyConfig: jetty-config.xml
workflowRefreshPeriod: 1
processingCleanerPeriod: 1
```

7.2.21.2.2 Fichier `version.conf`

```
binaryDataObjectVersions:
- BinaryMaster
- Dissemination
- Thumbnail
- TextContent
physicalDataObjectVersions:
- PhysicalMaster
- Dissemination
```

7.2.21.2.3 Fichier `storage-client.conf`

```
serverHost: {{ vitam_storageengine.host }}
serverPort: {{ vitam_storageengine.port_service }}
```

7.2.21.3 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

7.2.22 Security internal

7.2.22.1 Présentation

Rôle :

- Service d'authentification par certificats

Fonctions :

- Authentification par certificats
- Authentification personae

7.2.22.2 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/security-internal`.

7.2.22.2.1 Fichier `security-internal.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: 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 }}
```

7.2.22.3 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

7.2.23 siegfried

7.2.23.1 Présentation

Siegfried est un outil permettant la détection de format d'un fichier.

7.2.23.2 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*.

7.2.23.3 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

7.2.24 storage-engine

7.2.24.1 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

7.2.24.2 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/storage-engine`.

7.2.24.2.1 Fichier `driver-location.conf`

```
driverLocation: {{vitam_folder_lib}}
```

7.2.24.2.2 Fichier `driver-mapping.conf`

```
driverMappingPath: {{vitam_folder_data}}/  
delimiter: ;
```

7.2.24.2.3 Fichier `static-offer.json`

```
{% if vitam.storageofferdefault.https_enabled==true %}  
  {% set protocol = 'https' %}  
{% else %}  
  {% set protocol = 'http' %}  
{% endif %}  
[
```

(suite sur la page suivante)

(suite de la page précédente)

```
{% 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 }}",
  "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 %}
]
```

7.2.24.2.4 Fichier static-strategy.json

```
{
  "id" : "default",
  "hot" : {
    "copy" : {{ vitam_strategy|length }},
    "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 not loop.
↪last %}, {% endif %}
      {% endfor %}
    ]
  }
}
```

7.2.24.2.5 Fichier storage-engine.conf

```
urlWorkspace: {{ vitam.workspace | client_url }}
timeoutMsPerKB: 100
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 }}
#Basic Authentication
adminBasicAuth:
```

(suite sur la page suivante)

```
- userName: {{ admin_basic_auth_user }}
  password: {{ admin_basic_auth_password }}
```

7.2.24.3 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

7.2.25 offer

7.2.25.1 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

7.2.25.2 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/offer`.

7.2.25.2.1 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 ==
→ 'localhost' %}
- 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 }}
```

7.2.25.2.2 Fichier default-storage.conf

```
provider: {{ vitam_offers[offer_conf]["provider"] }}
{% if vitam_offers[offer_conf]["provider"] != "openstack-swift" %}
storagePath: {{vitam_folder_data}}
{% endif %}
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("") }}
swiftTrustTore: {{vitam_folder_conf}}/truststore_{{ vitam_struct.vitam_component }}.
→jks
swiftTrustTorePassword: {{ password_truststore }}
```

L'arborescence de stockage des fichiers dans l'offre est décrite dans le *DAT*.

7.2.25.3 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

7.2.26 worker

7.2.26.1 Présentation

Ce composant permet de réaliser l'ensemble des traitements sur les archives.

Rôle :

- Fournir un moyen d'exécuter les traitements sur les archives, piloté par le composant processing.

7.2.26.2 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`.

7.2.26.2.1 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
```

7.2.26.2.2 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 }}
```


7.2.26.2.3 Fichier metadata-client.conf

Ce fichier permet de définir l'accès au metadata.

```
serverHost: {{ vitam.metadata.host }}
serverPort: {{ vitam.metadata.port_service }}
```

7.2.26.2.4 Fichier storage-client.conf

Ce fichier permet de définir l'accès au storage.

```
serverHost: {{ vitam.storageengine.host }}
serverPort: {{ vitam.storageengine.port_service }}
```

7.2.26.2.5 Fichier version.conf

```
binaryDataObjectVersions:
- BinaryMaster
- Dissemination
- Thumbnail
- TextContent
physicalDataObjectVersions:
- PhysicalMaster
- Dissemination
```

7.2.26.2.6 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
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 %}
```

Paramètres obligatoires :

- **processingUrl** : URL de connexion au composant 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 appartient en fonction de tâche exécutée
- **capacity** : capacité du worker en mode parallèle de tâche (par défaut à 1 dans l'ansible, si non définie)

7.2.26.3 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

7.2.27 workspace

7.2.27.1 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

Fonctions :

- Utilisation du moteur de stockage dans un mode minimal (Opérations CREATE, READ, DELETE sur 1 seule offre de stockage)

7.2.27.2 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`.

7.2.27.2.1 Fichier workspace.conf

```
storagePath: {{vitam_folder_data}}
jettyConfig: jetty-config.xml
provider: filesystem
```

7.2.27.3 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

Intégration d'une application externe dans Vitam

8.1 Prérequis

L'application externe devra être en mesure de requêter les composants *VITAM* ingest-external et access-external sur leur port de service 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.

8.2 Intégration de certificats clients de VITAM

8.2.1 Authentification applicative SIA

Le certificat client de l'application externe devra 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.

- Editer le fichier `environments/group_vars/all/vitam_security.yml` et ajouter le(s) entrée(s) supplémentaire(s) (sous forme répertoire/fichier.crt) dans la directive `admin_context_certs`
- Déposer le certificat client de l'application dans `environments/certs/client-external/clients/external`

Avvertissement : Bien vérifier qu'aucun certificat ou CA non souhaité ne soit présent dans <code>environments/certs/*</code>

Lorsque l'authentification du client, par le protocole TLS, est réalisée, l'AC du certificat client doit être déployée dans le truststore des composants *VITAM* ingest-external et access-external. 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 nécessaire d'inclure l'AC associée dans le truststore des composants *VITAM* ingest-external et access-external.

Vitam fournit une arborescence de répertoire permettant d'ajouter automatiquement la chaîne de certification client dans les bons keystores.

- Déposer les certificats de la chaîne de certification client dans *environnements/certs/client-external/ca*. Ensuite, si ce n'est pas déjà fait, il reste à positionner les autres certificats, puis générer les keystores. Pour cela, se référer au *DIN*, chapitre concernant la gestion des certificats.

8.2.2 Authentification *personae*

Les exemple suivants permettent d'ajouter ou supprimer un certificat présent sous */path/to/certificate*.

8.2.2.1 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'
```

8.2.2.2 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'
```

8.3 Déploiement des keystores

8.3.1 Vitam n'est pas encore déployé

Déployer Vitam en suivant la procédure indiquée dans le *DIN*.

8.3.2 Vitam est déjà déployé

Suivre la procédure de la section *Mise à jour des certificats* (page 8).

9.1 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*.

9.1.1 Etat par Consul

Se connecter à l'IHM de Consul et recenser les états des composants de la solution logicielle *VITAM*.

Services

Service	Node Health	Tags
access-external	✓ 4	
access-internal	✓ 4	
cerebro	✓ 3	
consul	✓ 6	
elastic-kibana-interceptor	✓ 4	
elasticsearch-data	✓ 12	
elasticsearch-log	✓ 12	
functional-administration	✓ 4	
ihm-demo	✓ 4	
ihm-recette	✓ 4	
ingest-external	✓ 6	
ingest-internal	✓ 4	

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.`

9.1.2 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.

9.2 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
```

Questions Fréquemment Posées

10.1 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.

10.2 Retour d'expérience / cas rencontrés

10.2.1 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 :

- Centos¹⁹
- Debian²⁰

10.2.2 Mongo-express ne se connecte pas à la base de données associée

Si mongoDB a été redémarré, il faut également redémarrer mongo-express.

10.2.3 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

<https://www.rsyslog.com/rhelcentos-rpms/>
<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 officielle²¹.

10.2.4 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](#)²²). 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.

10.2.5 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`²³ et `mongotop`²⁴.

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 troubleshoot **les problèmes dus à la réplication**²⁵ :

```
mongo --host <ip_service> --port 27019 --username vitamdb-localadmin --password
↳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**²⁶) :

```
echo "{nThreads:16,fileSizeMB:10000,r:true,w:true}" | mongoperf
```

10.2.6 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/>

10.2.7 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 fonctionnal-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.

10.2.8 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.

Les sections qui suivent donnent une description plus fine pour l'exploitation des services VITAM.

11.1 Access

11.1.1 Introduction

Cette documentation permet de paramétrer le port, le host ... à l'environnement de dev, d'intégration et de production pour le module access client et access-rest

```
#Configuration access-client serverHost : localhost serverPort : 8189
#Configuration access-rest urlMetaData : http://localhost:8088
#Configuration du serveur jetty jettyConfig : access-jetty.xml
```

11.2 Batch-Report

11.2.1 Présentation

Le composant batch-report permet de stocker des données de traitements de masse (en particulier, élimination) pour les agréger 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 agrégées sous forme de rapports.

11.2.2 batch-report-exploitation

Ce document spécifie la configuration (fichiers de config) pour lancer le services de batch-report.

1. Serveur batch-report :

Pour lancer le serveur batch-report, un fichier de config suivant est nécessaires :

- `batch-report.conf` : Contient la configuration du serveur MongoDB, du serveur jetty, les tenants.
- `jetty-config.xml` : contenant le information pour lancer le serveur jetty de batch-report. Ce fichier de jetty précise aussi la configuration TSL du mode SSL du serveur : les keystores et password pour les load, les algorithmes de chiffrement supportés ...

2. 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 ou le client se connectent pour les requêtes.

Tous ces fichiers de configuration seront mis dans le répertoire `/vitam/conf/batch-report`.

11.3 Common

11.3.1 Présentation

11.3.2 Format Identifiers

Les services d'identification de formats peuvent être déployés sur tous les serveurs applicatifs vitam.

11.3.2.1 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-exister sur le serveur 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

11.4 Functional administration

11.4.1 Présentation

11.4.2 Functional administration

11.4.2.1 Configuration du Functional administration

functional-administration.conf : Fichier Yaml de configuration du serveur *worker*. Il possède une propriété :

- **listMinimumRuleDuration** : la durée minimum de chaque type de règle par tenant

```
listMinimumRuleDuration:
  2:
    AppraisalRule : 1 year
```

11.5 Ingest

11.5.1 Introduction

Ce document présente les configs pour utiliser des modules différents de ingest. Deux configurations à voir pour ingest-external et ingest-internal

11.5.2 ingest-external-exploitation

Ce document spécifie la configurations (fichiers de config) pour lancer les services de ingest-external.

1. Serveur ingest-external :

Pour lancer le serveur ingest-external, deux fichier config suivant sont nécessaires : - ingest-external.conf : préciser le répertoire temporaire ou les fichiers temporaires stockés et la configuration du serveur jetty. - jetty-config.xml : contenant le information pour lancer le serveur jetty de ingest-external. Ce fichier de jetty précise aussi la configuration TLS du mode SSL du serveur : les keystores et password pour les load, les algorithmes de chiffrement supportés ... - scan-clamav.sh : le script de scan pour détecter les virus qui sera appelé par les service ingest-external. -shiro.ini : la configuration de shiro permettant de filtrer des requêtes côté serveur. Il décrit différents paramètres : le marcher ; le filter et les keystores utilisé pour le filtre (truststore.jks et granted_certs.jks) - le répertoire de keystore tls : ce répertoire contient différents fichiers de keystore permettant de lancer le serveur ingest-external en mode SSL.

- **truststore.jks** : contenant les CA qui attribue des certificats

- **keystore.jks** : contenant la clé privé et le certificat du serveur
- **granted_certs.jks** : contenant les certificat des clients qui auront une authentification

1. Client ingest-external :

Pour la création d'un lient ingest-external, nous avons besoin aussi le fichier de configuration ingest-external-client.conf qui précise le serveur host et la porte du serveur ou le client se connectent pour les requêtes.

Tous ces fichiers de configuration seront mis dans le répertoire /vitam/config. Les exemples de ces fichiers se trouvent dans les répertoires de src/test/resources correspondants.

11.5.3 ingest-internal-exploitation

Ce document spécifie la configurations (fichiers de config) pour lancer le services de ingest-internal.

1. Serveur ingest-internal : Pour lancer le serveur ingest-internal, deux fichier config suivant sont nécessaires - ingest-internal.conf : préciser les urls de services Processing & Workspace, et la configuration du serveur jetty. - jetty-config.xml : contenant le information pour lancer le serveur jetty de ingest-internal.

1.Client ingest-internal : Pour la création d'un lient ingest-internal, nous avons besoin aussi le fichier de configuration ingest-internal-client.conf qui précise le serveur host et la porte du serveur ou le client se connecte pour les requêtes.

Tous ces fichiers de configuration seront mis dans le répertoire /vitam/config. Les exemples de ces fichiers se trouvent dans les répertoires de src/test/resources correspondants.

11.6 Security-Internal

11.6.1 Introduction

Ce document présente la configuration pour le module security-internal.

11.6.2 security-internal-exploitation

Ce document spécifie la configuration (fichiers de config) pour lancer le services de security-internal.

1. Serveur security-internal :

Pour lancer le serveur security-internal, deux fichier config suivant sont nécessaires :

- **security-internal.conf** : Contient 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.
- **jetty-config.xml** : contenant le information pour lancer le serveur jetty de security-internal. Ce fichier de jetty précise aussi la configuration TSL du mode SSL du serveur : les keystores et password pour les load, les algorithmes de chiffrement supportés ...
- **personal-certificate-permissions.conf** : Configuration des permissions nécessitant une authentification personae ou ne nécessitant pas d'authentification personae.

2. Client security-internal :

Pour la création d'un client security-internal, nous avons besoin aussi le fichier de configuration internal-security-client.conf qui précise le serveur host et la porte du serveur ou le client se connectent pour les requêtes.

Tous ces fichiers de configuration seront mis dans le répertoire /vitam/config. Les exemples de ces fichiers se trouvent dans les répertoires de src/test/resources correspondants.

11.7 Logbook

11.7.1 Présentation

11.7.2 Logbook Exploitation

11.7.2.1 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 règle 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)

11.8 Metadata

11.8.1 Présentation

11.9 Processing

11.9.1 Introduction

11.9.1.1 But de cette documentation

Le but de cette documentation est d'expliquer la configuration et l'exploitation de ce module.

11.9.2 Processing

Nom de l'image docker : **processing**

Dans cette image est déployé le module processing

11.9.2.1 Configuration du worker

Dans `/vitam/conf` :

1. **processing.conf** : Fichier Yaml de configuration du server *processing*. Il possède une propriété :
 - **jettyConfig** : emplacement du fichier 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

11.9.2.2 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`

11.10 Storage

11.10.1 Introduction

11.10.1.1 But de cette documentation

Le but de cette documentation est d'expliquer la configuration et l'exploitation des modules :

- **storage-engine**
- **storage-offer-default**

11.10.2 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`)

11.10.2.1 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 DriverManager (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 persistance 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'authentification. 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

11.10.2.2 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 persistance. 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.

11.10.2.3 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

11.10.3 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.

11.10.3.1 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 objets (/vitam/data).
2. **server-identity.conf** : identification du serveur
3. **logback.xml** : configuration des logs

11.10.3.2 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

11.11 Technical administration

11.11.1 Présentation

11.12 Worker

11.12.1 Introduction

11.12.1.1 But de cette documentation

Le but de cette documentation est d'expliquer la configuration et l'exploitation de ce module :

- **worker**

11.12.2 Storage Engine

Nom de l'image docker : **worker**

Dans cette image est déployé le module worker

11.12.2.1 Configuration du worker

Dans `/vitam/conf` :

1. **worker.conf** : Fichier Yaml de configuration du server *worker*. Il possède une propriété :
 - **jettyConfig** : emplacement du fichier de configuration XML *jetty* (exemple `jetty-config.xml`)
 - **registerServerHost** : le nom d'hôte du serveur courant auquel le client worker chargé par le processing va se connecter (Exemple : `localhost`)
 - **registerServerPort** : le port du serveur courant auquel le client worker chargé par le processing va se connecter (Exemple : `8082`)
 - **processingUrl** : URL d'accès au service distant *processing* (exemple `http://localhost:8088`)
 - **urlWorkspace** : URL d'accès au service distant *workspace* (exemple `http://localhost:8088`)
 - **urlMetadata** : URL d'accès au service distant *metadata* (exemple `http://localhost:8088`)
 - **classificationLevel** : *allowList* : Les niveaux de classifications acceptés par la plateforme *authorizeNotDefined* : S'il est true, la plateforme accepte les AU sans niveau de confidentialité. S'il est false, la plateforme refuse les AU sans niveau de confidentialité.
2. **version.conf** : Fichier contenant la liste des version valides pour les SEDA. Il contient deux listes : une pour « `binaryDataObjectVersions` », une pour « `physicalDataObjectVersions` ».
3. **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
4. **storage-client.conf** : Fichier de configuration du client qui communique avec le **storage-engine**. Il contient les propriétés suivantes :
 - **serverHost** : host distant du service storage-engine
 - **serverPort** : port distant du service storage-engine
5. **server-identity.conf** : identification du serveur
6. **logback.xml** : configuration des logs

11.12.2.2 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>/worker/v1/status`

11.13 Workspace

11.13.1 Présentation

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 : régénération 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	Us-ages	Origine	Création	Sup-pression	Renouvellement
Interne	CA	ingest & access	PKI partenaire	<i>proc. par défaut</i>	<i>proc. par défaut</i>	<i>proc. par défaut</i>
		offer	PKI partenaire	<i>proc. par défaut</i>	<i>proc. par défaut</i>	<i>proc. par défaut</i>
	Certif	Horo-datage	PKI partenaire	<i>proc. par défaut</i>	<i>proc. par défaut</i>	<i>proc. par défaut</i>
		Storage (swift)	Offre de stockage	<i>proc. par défaut</i>	<i>proc. par défaut</i>	<i>proc. par défaut</i>
		ingest	PKI partenaire	<i>proc. par défaut</i>	<i>proc. par défaut</i>	<i>proc. par défaut</i>
		access	PKI partenaire	<i>proc. par défaut</i>	<i>proc. par défaut</i>	<i>proc. par défaut</i>
		offer	PKI partenaire	<i>proc. par défaut</i>	<i>proc. par défaut</i>	<i>proc. par défaut</i>
		Times-tamp	PKI partenaire	<i>proc. par défaut</i>	<i>proc. par défaut</i>	<i>proc. par défaut</i>
IHM demo	CA	ihm-demo	PKI partenaire	<i>proc. par défaut</i>	<i>proc. par défaut</i>	<i>proc. par défaut</i>
	Certif	ihm-demo	PKI partenaire	<i>proc. par défaut</i>	<i>proc. par défaut</i>	<i>proc. par défaut</i>
SIA	CA	Appel API	PKI distante	proc. par défaut (PKI distante)	<i>proc. par défaut</i>	proc. par défaut (PKI distante) + recharger Certifs
	Certif	Appel API	PKI distante	Génération + copie répertoire + deploy (par la suite, appel API d'insertion)	Sup-pression Mongo	Suppression Mongo + API d'insertion
Personae	Certif	Appel API	PKI distante	API ajout	API sup-pression	API suppression + API ajout

Remarques :

- Lors d'un renouvellement de CA SIA, il faut s'assurer que les certificats qui y correspondaient sont retirés de MongoDB et que les nouveaux certificats sont ajoutés par le biais de l'API dédiée.
- Lors de toute suppression ou remplacement de certificats SIA, s'assurer que la suppression / remplacement des contextes associés soit également réalisée.
- 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.]
 - Seule la dernière version majeure continue de bénéficier de nouvelles versions mineures.
- **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 patches.

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 patches 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 patches sont ensuite inclus dans une version bugfix ultérieure.

12.2.3 Gestion des patches

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 patches 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 patches 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

1	Vue d'ensemble d'un déploiement <i>VITAM</i> : zones, composants	6
---	--	---

Liste des tableaux

1	Documents de référence VITAM	2
---	--	---

A

API, 2

B

BDD, 2

C

CA, 2

COTS, 3

D

DAT, 3

DEX, 3

DIN, 3

DNSSEC, 3

DUA, 3

I

IHM, 3

J

JRE, 3

JVM, 3

M

MitM, 3

N

NoSQL, 3

O

OAIS, 3

P

PCA, 3

PDMA, 3

PKI, 3

PRA, 3

R

REST, 3

RPM, 3

S

SAE, 3

SEDA, 3

SIA, 3

SIP, 3

T

TNR, 3

V

VITAM, 3