



Chantier Technique 4 du 17/02/2016

Modèles approfondis d'implémentation

Profils concepteur d'application

Valeurs Immatérielles Transférées aux Archives pour Mémoire

Aperçus des chantiers techniques 1/2

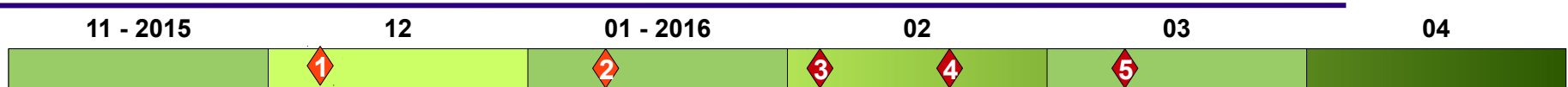
Vision non limitative

- Quelle infrastructure pour Vitam ?
 - Quels matériels, OS, socles logiciels ?
 - Quels stockages ?
 - Quelles implantations ?
 - Quel PRA / PCA ?
- Quelle intégration de Vitam dans mon SI ?
 - Intégration fonctionnelle
 - *hors scope (voir chantiers fonctionnels)*
 - Intégration de l'exploitation
 - *Comment superviser ?*
 - *Comment sauvegarder / restaurer ?*
 - Intégration applicative : API
 - *Application frontale (Front-Office) via les API Vitam*

Objectif transverse : Quelles clauses dans un marché public pour la réalisation de l'intégration (2016 : API, 2018 : Stockage, IaaS) ?

Aperçus des chantiers techniques 2/2

Premiers chantiers envisagés



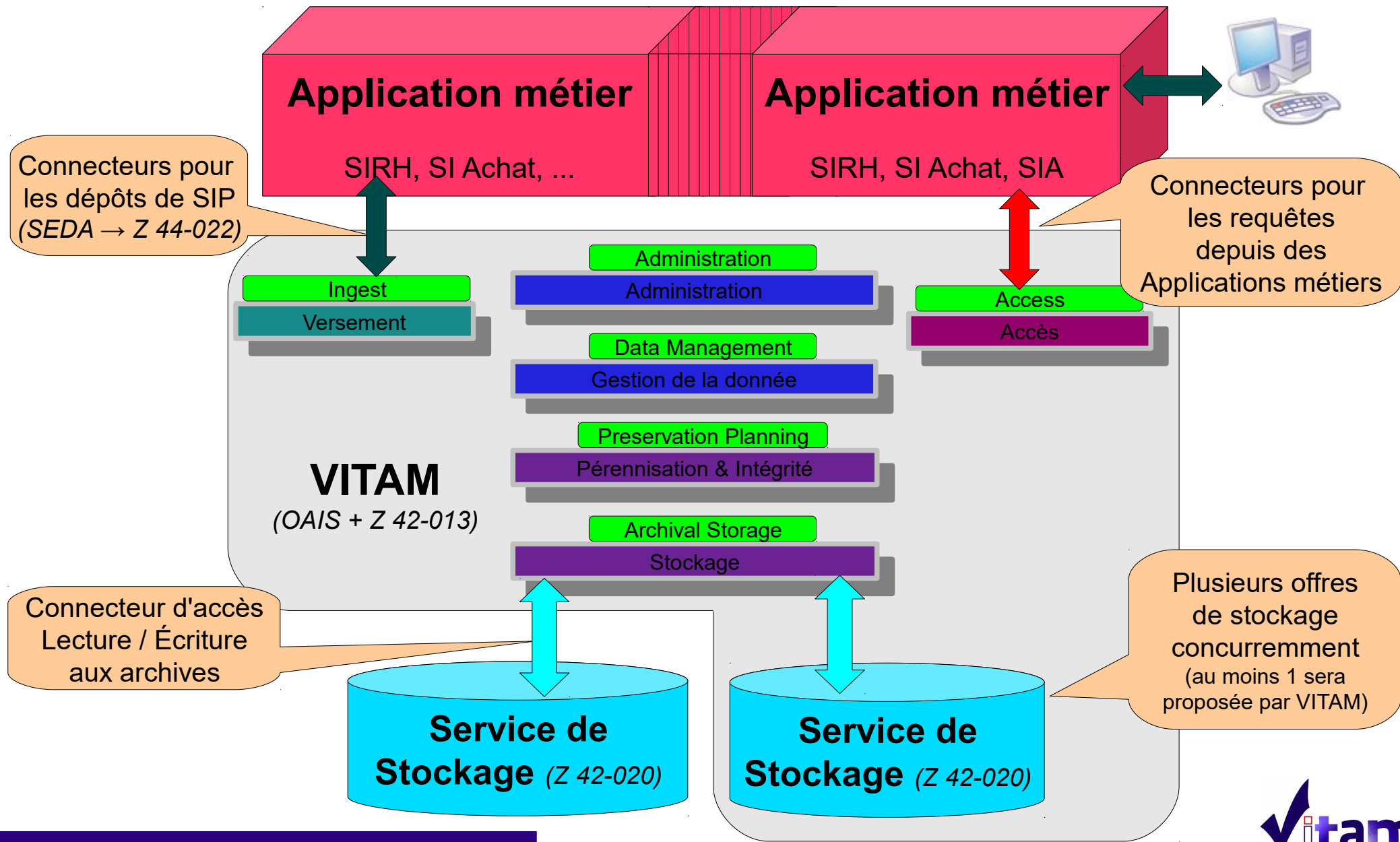
- **09/12/2015 : Chantier 1)** Infrastructures actuelles : profils exploitation, architecture technique
 - Infrastructure : Outils de déploiement (Virtualisation, OS), Stockage
 - Exploitation : Supervision, Sauvegarde/Restauration, PRA/PCA
 - Outils de transfert de fichiers
- **12/01/2015 : Chantier 2)** RETEX et préconisations existantes : profils concepteur d'application, architecture applicative
 - Outils de transfert de fichiers
 - API REST et Authentification d'application à application
 - Langage de programmation pour l'usage de bibliothèques Vitam
- **XX/02/2015 Chantier 3)** Contraintes de sécurité : profils architecture de production, RSSI
 - Zoning, Multi-sites, PCA/PRA, Déploiement, Multi-tenants
 - Protocoles, Authentification
 - Outils (Antivirus, Pare-feu, sondes, ...)
 - Logs et traces, anonymisation dans le cadre du support
- **XX/02/2015 Chantier 4)** Modèles approfondis d'implémentation : profils concepteur d'application
 - Modèles d'API par usage (versement, accès, administration), Outillage
 - Modèle de Déploiement de Vitam
- **XX/03/2015 Chantier 5)** Modèles approfondis d'implémentation : profils exploitation, architecture technique
 - Prise en compte de l'adhérence avec les contextes d'exploitation et de sécurité
 - Modèles de sauvegarde, Modèle de stockage
 - Modèle de déploiement, PaaS

Avez vous d'autres suggestions de sujets techniques ?

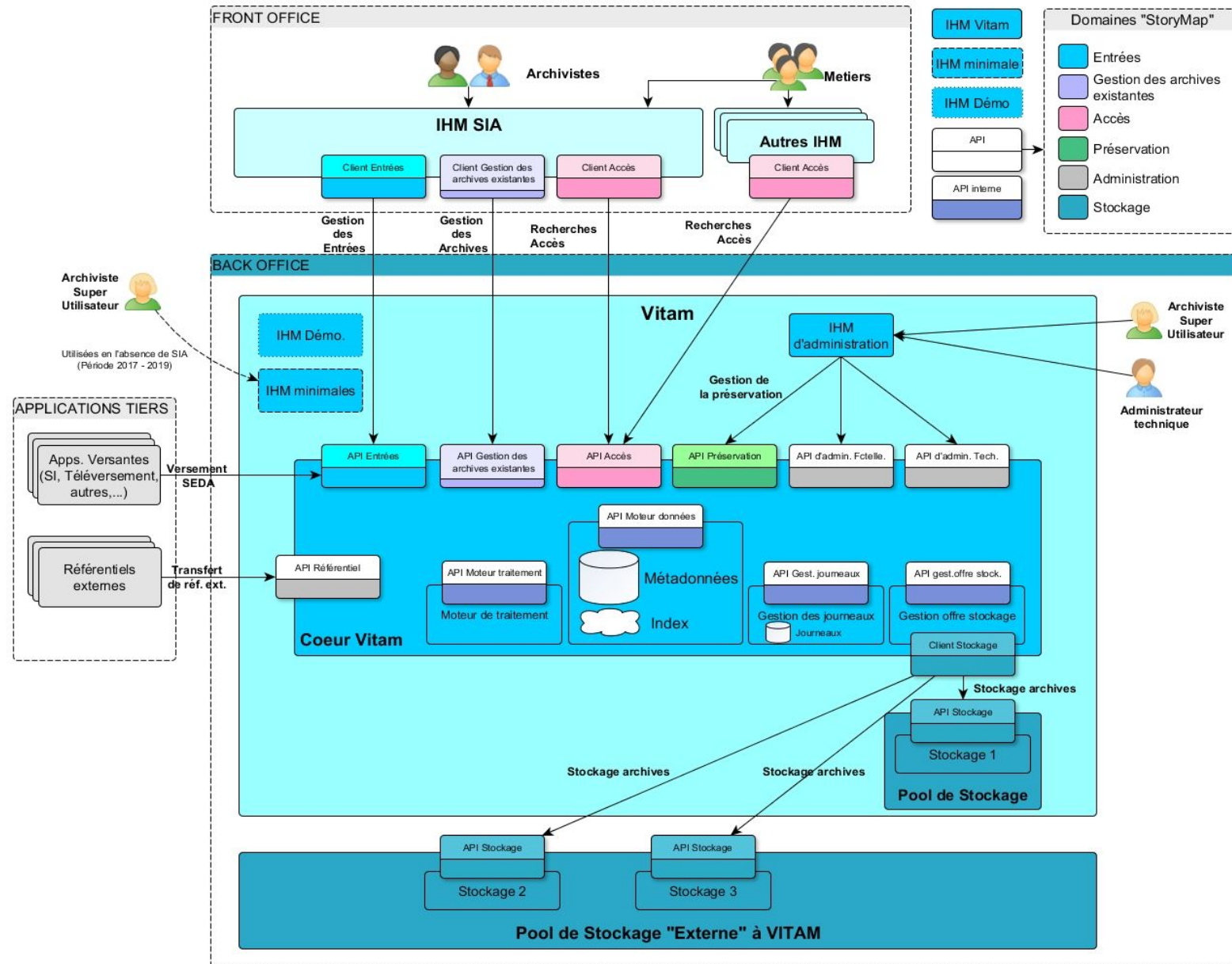
- Modèles approfondis d'implémentation
 - Modèle général
 - Modèles d'API par usage
 - *Authentication et Multi-tenants*
 - *Modèle de requêtes : Domain Specific Language*
 - *Tâches asynchrones*
 - *Modèle de données*
 - *Accès*
 - *Versement*
 - *Gestion*
 - Outillage
 - Exemples (annexe)

L'architecture générale de la solution logicielle Vitam

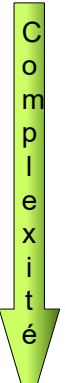
Interfaçage, Indépendance, Réutilisation, Sécurité



L'architecture fonctionnelle de la solution logicielle Vitam



- Modèles approfondis d'implémentation
 - Modèle général
 - Modèles d'API par usage
 - *Authentification et Multi-tenants*
 - *Modèle de requêtes : Domain Specific Language*
 - *Tâches asynchrones*
 - *Modèle de données*
 - *Accès*
 - *Versement*
 - *Gestion*
 - Outillage
 - Exemples (annexe)

- L'objet est ici principalement sur une prévue des API
 - Les End-points principaux
 - Les expressions de requêtes et réponses
 - L'objectif est d'essayer d'obtenir une API
 - Compréhensible
 - Simple à utiliser
 - Simple à interfacer avec une application existante
 - Mais aussi de voir les aides qui peuvent être apportées pour le développement des inter-connexions
- 
- *Documentation*
 - *Exemple de codes*
 - *Framework de génération de requêtes et de parsing de réponses*
 - *Framework de connexions (protocole HTTPS)*

- Responsabilités de l'application Front-office
 - Ne pas donner accès « directement » à Vitam : en rupture de flux
 - *Gestion des droits fins d'accès*
 - *Front-offices seuls autorisés à accéder à Vitam, pas les navigateurs*
- End-Point
 - Respect du modèle REST Full
- Requête
 - Exprimé dans le Body via la définition d'un langage spécifique (Domain Specific Language)
 - Exprimable pour certains End-Point sur des cas simples dans l'URI
 - *Mais sera toujours aussi exprimable dans le Body*
- Réponse
 - Exprimé dans le Body mais aussi dans le Header
- Authentification
 - CF propositions séance n°2 : SSL/TLS + BASIC avec possibilité d'externalisation
 - *Externalisation pour bénéficier de OAUTH2, Certificats, autres méthodes*
- Tâches asynchrones
- Multi-tenants

Modèle général : End-Points

- **Versionning des API**
 - Domaine = nom DNS du service
 - Application = domaine de l'application (versement, accès, gestion, ...)
 - Version = v1, v2, ... versions majeures uniquement (version fine dans le header)

<https://domaine/application/version/>
- **Accès à une ressource/collection**

<https://domaine/application/version/ressources>
- **Accès à un élément de cette ressource**

https://domaine/application/version/ressources/identifiant_ressource
- **Accès à une deuxième ressource issue d'un élément**

https://domaine/application/version/ressources/identifiant_ressource/ressources2
- **Commandes :**
 - GET : Lecture
 - POST : Création
 - PUT : Mise à jour complète
 - PATCH : Mise à jour partielle, modification d'un état (nécessité mais PUT possible)
 - DELETE : Effacement, Annulation
 - HEAD : Informations succinctes (a priori non nécessaire)
 - OPTIONS : Opérations permises (a priori non nécessaire)

Modèle général : End-Points

- Services transverses offerts par chaque « application »
<https://domaine/application>
 - Pour les API externes
 - *API de vérification du statut du module*
 - *API permettant de consulter la version du service et la version de l'application déployée*
 - Pour les API internes (non visibles), en plus
 - *API de vérification de l'état des dépendances directes du module*
 - *API d'activation/désactivation d'un service*
 - *API de récupération des métriques du module*
 - *Métriques système et JVM*
 - *Métriques d'utilisation des API (par version)*
 - » *Cette fonctionnalité pourrait remonter pour chaque « endpoint » :*
 - » *le nombre d'appels,*
 - » *les temps d'exécution maximum/minimum/moyen/etc.*
 - » *le nombre d'erreurs rencontrées*
 - » *etc.*
 - » *Le tout sur différentes fenêtres temporelles.*
 - *API de sauvegarde (export) et de restauration (import) (interne)*

- Arguments
 - Cas simples (a priori exclu) : dans l'URL (<https://domaine/application?arguments>)
 - Cas complexes : dans le body HTTP
- Compatibilité multi-framework
 - Toutes les commandes ne sont pas implémentées, ni toutes les possibilités HTTP comme le body avec un GET
 - Header : « *X-HTTP-Method-Override: GET* » permet de prendre en compte une requête « GET virtuelle » avec Body en utilisant un « POST »
- Codes retours : standard HTTP
 - Quelques codes
 - 200 : OK, 201 : créé, 202 : accepté (requête asynchrone)
 - 204 : OK sur un effacement, 206 : réponse partielle (pagination)
 - 302 : API dépréciée, 410 : API obsolète, 501 : API non encore disponible
 - 400 : requête incorrecte, 401 : non authentifié, 403 : non autorisé
 - 404 : non trouvé
 - 429 : fréquence d'appels trop élevée
 - 503 : Service indisponible

- Modèles approfondis d'implémentation
 - Modèle général
 - Modèles d'API par usage
 - *Authentification et Multi-tenants*
 - *Modèle de requêtes : Domain Specific Language*
 - *Tâches asynchrones*
 - *Modèle de données*
 - *Accès*
 - *Versement*
 - *Gestion*
 - Outillage
 - Exemples (annexe)

- Modèles approfondis d'implémentation
 - Modèle général
 - Modèles d'API par usage
 - *Authentication et Multi-tenants*
 - *Modèle de requêtes : Domain Specific Language*
 - *Tâches asynchrones*
 - *Modèle de données*
 - *Accès*
 - *Versement*
 - *Gestion*
 - Outillage
 - Exemples (annexe)

Protocoles, Authentications

- Protocoles d'échanges entre le SI et Vitam
 - Protocoles envisagés
 - *HTTPS*
 - *Autres protocoles pour des flux de fort dimensionnement (FTPS, WAARP, ...)*
 - Selon les protocoles, différents mécanismes d'authentications peuvent être mis en œuvre
 - *Clef (côté serveur Vitam, côté client Vitam) pour la partie SSL*
 - *User/Password*
 - *Mécanisme SI pour la partie HTTPS ?*
 - *Un connecteur devra permettre l'utilisation d'authentications liées au SI (cf Digest Scheme, JSON Web Token, ...)*
 - *A noter RGI : Recommandation OAUTH2 mais très orienté utilisateur transmis entre 2 applications, ce qui n'est pas notre cas*

Protocoles, Authentification Grandes familles, complémentaires ?

	Avantages	Inconvénients
Challenge (ex : Digest)	Pas de tiers de confiance nécessaire Pas de risque de rejeu	Nécessite 2 requêtes pour faire l'authentification Nécessite de stocker le mot de passe/secret sur le serveur Dans le cas de Vitam : Nombre de secrets à gérer Risque de Man-In-The-Middle si le client peut gérer des méthodes d'authentification dégradées (ex : Basic) / Complexe à mettre en œuvre proprement
Scellement (ex : HMAC)	Authentification et requête dans la même requête HTTP (stateless) Pas de tiers de confiance nécessaire	Nécessite de stocker le mot de passe/secret sur le serveur Sensible au rejeu (nécessite de définir une date d'expiration du sceau) Nombre de secrets à gérer
Certificat client TLS	Une révocation de la chaîne de confiance est prise en compte immédiatement par le système Apporte une authentification « plus forte » (clé privée + passphrase de la clé privée) Pas de risque de rejeu	Complexe à mettre en œuvre (chaîne de confiance, vérification des CRL) et à maintenir Une révocation d'une AC intermédiaire peut provoquer un incident de production
Systèmes basés sur un fournisseur d'identité (Oauth2, SAML, Keystone pour Openstack)	Gestion centralisée de l'identité de la plateforme	Complexité de mise en œuvre Le fournisseur d'identité est un composant central critique/SPOF dans les échanges

Des règles ou recommandations existantes dans vos SI ?

Protocoles, Authentications

Challenge et Scellement = Authorization: OAuth realm, Digest Scheme

Challenge sous la forme de Digest

Côté Client

```
request.headers['Client-Id'] = getMyId()
String digest = hmacSha256(request, password)
String val = 'Foo ' + digest
Request.headers['Authorization'] = val
send(request)
```

Exemple : OAuth 1.0a

```
Authorization: OAuth realm='http://app.example.com',
oauth_consumer_key='xxxxxxxxx',
oauth_token='xxxxxxxxxxx',
oauth_signature_method='HMAC-SHA1',
oauth_signature='xxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
oauth_timestamp='123456789',
oauth_nonce='xxxxxxxxxxxxxxxxxxxxxxxxxxxxx'
```

Côté Serveur

```
String clientId = request.headers['Client-Id']
Byte[] password = lookupPasswordClient(clientId)
String serverComputeDigest = hmacSha256(request, password)
String val = request.headers['Authorization']
String clientSpecifiedDigest = val.remove('Foo ')
If (clientSpecifiedDigest != serverComputeDigest) (
    sendError(401, response)
    return
)
// Sinon requête authentifiée
```

Avantages :

- Probablement le plus sécurisé
- Password jamais transmis
- Garantie de l'authenticité du message (ce que ne fait pas HTTPS)
- Non susceptible d'attaque « Man In The Middle »

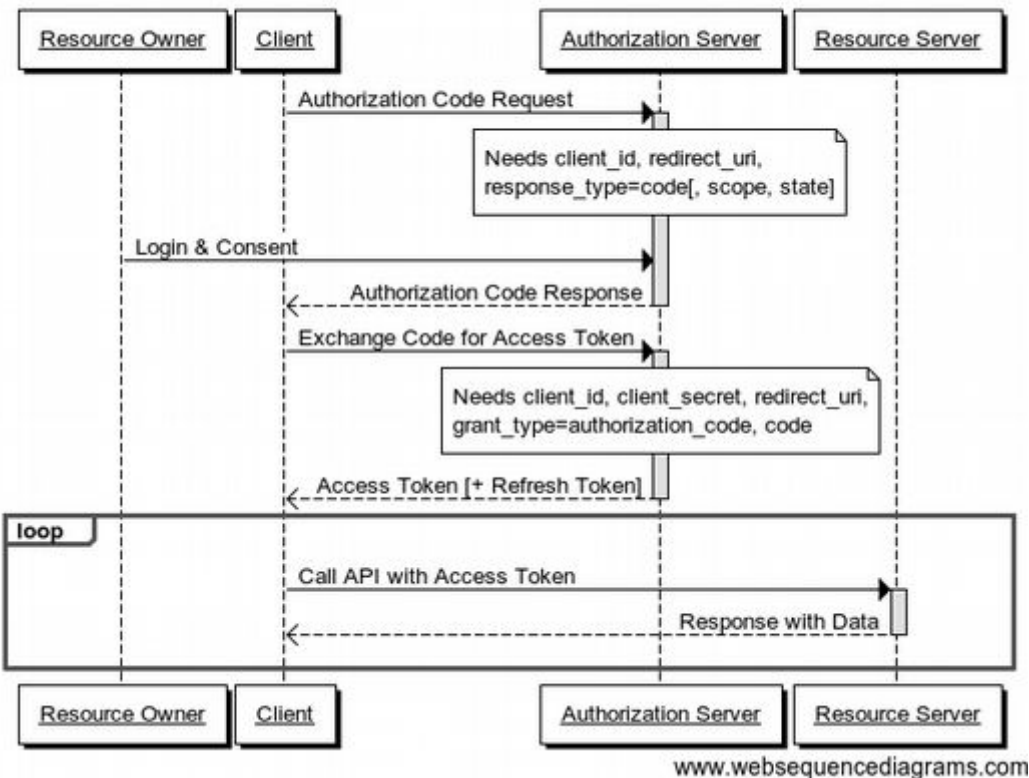
Inconvénients :

- Difficile à implémenter de manière sûre
- Difficile à comprendre et utiliser
- Difficile d'implémenter des bibliothèques
- Le client doit avoir un password stable

Protocoles, Authentications

Fournisseur d'identité : Authorization: OAUTH2

Authorization Code Grant Flow



Avantages :

- Plus facile à utiliser qu'un Digest
- Un standard de fait pour le format du token (JWT)
- Peut contenir un état – pas de session serveur
- Ne nécessite pas l'accès constant au password utilisateur

Inconvénients :

- HTTPS requis
- Ne garantit pas l'authenticité du message (comme Digest le peut)
- Sujet à attaques « Man In The Middle »
- La création de Token et le renouvellement peut être compliqué et confus
- Le contenu des Token n'est pas standardisé

Protocoles, Authentications

Fournisseur d'identité : Token Authorization: Bearer, Json Web Token

JWT est une chaîne encodée en Base64

```
XXXXXXXXXXXXXXXXXX, ← Header  
  
YYYYYYYYYYYYYYYY, ← Body (Claims)  
ZZZZZZZZZZZZZZZZ ← Cryptographic Signature
```

Une fois décodée, le contenu est :

```
{ 'typ' : 'JWT' ; 'alg' : 'HS256' }, ← Header  
{ 'iss' : 'http://trustyapp.com/',  
  'exp' : 1234567890,  
  'sub' : 'users/123456789',  
  'scope' : 'self api/buy'}, ← Body (Claims)  
{ 'tls' : 'xxxxxxxxxxxxxx' } ← Cryptographic Signature
```

JWT : la partie « claims » est la plus intéressante

```
{  
  'iss' : 'http://trustyapp.com/', ← Qui a fournit le token  
  'exp' : 1234567890, ← Quand il expire  
  'sub' : 'users/123456789', ← Qui il représente  
  'scope' : 'self api/buy' ← Ce qu'il peut faire  
}
```

```
GET /admin HTTP/1.1  
Authorization: Bearer xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

Avantages :

- Authenticité valide en raison de la signature
- Structuré, inter-opérable
- Peut contenir un état – pas de session serveur
- Ne nécessite pas l'accès constant au password utilisateur

Inconvénients :

- HTTPS requis
- L'authenticité est limitée dans le temps par la durée de validité du Token
- La création de Token et le renouvellement peut être compliqué et confus
- Attention au stockage des JWT (chiffrés) côté serveur

- Les aspects multi-tenants seront pris en compte selon 2 modalités
 - 1) Implicitement
 - L'application Front-office n'est utilisée que pour un seul tenant
 - Vitam ajoute systématiquement les informations à chaque requête du tenant concerné
 - 2) Explicitement
 - L'application Front-office utilise plusieurs tenants (fonction de l'utilisateur connecté)
 - L'application doit intégrer un code dans le Header pour indiquer le code du tenant concerné
 - *X-TenantId : valeur*
- La séparation est logique
 - Données : séparation par filtre
 - Stockage : séparation par container

- Modèles approfondis d'implémentation
 - Modèle général
 - Modèles d'API par usage
 - *Authentication et Multi-tenants*
 - *Modèle de requêtes : Domain Specific Language*
 - *Tâches asynchrones*
 - *Modèle de données*
 - *Accès*
 - *Versement*
 - *Gestion*
 - Outillage
 - Exemples (annexe)

Domain Specific Language Vitam

Règles générales

- Dans le body : langage de requête
 - DSL VITAM
 - *SQL : pas de plein texte, parser difficile*
 - *NoSQL : pas de norme*
 - *Abstraction indispensable (masquer l'implémentation)*
- Typographie
 - Snake : « `propriete_avec_multiple_noms` »
 - *Mais pas « `proprieteAvecMultipleNoms` »*
 - Variables protégées : préfixe = « `_` » en URL
 - *Exemple : « `_limit=10` »*
 - Body au format JSON
 - *Contient des informations spécifiques à la requête pour la collection*
 - *Peut contenir une « Query » (DSL)*
- Pagination
 - `_offset/_limit` en URL, `offset / limit` dans la Query
 - Range dans le Header pour les octets d'un fichier binaire)
- Tri
 - `_sort=champ, _desc=champ` en URL, `orderby` dans la Query

Domain Specific Language Vitam

Query dans le Body

- DSL : Un langage (JSON), Logique SQL => NoSQL
Projections, Collections, Requêtes (critères=query), Filtres (tri, limite)
SELECT field1, field2 FROM table WHERE field3 < value LIMIT n SORT field1 ASC
- Modèle générique CRUD
 - Create = POST
 - *data : { champ : valeur, champ : { champ : valeur } }*
 - Read = GET
 - *filter : { limit, offset, orderby }, projection : { field : 0/1, ... }*
 - Update = PUT (avec forme ~ POST) / PATCH
 - *action : { set : { field : value, ... }, inc : { field : value }, ... }*
 - Delete = DELETE
 - *filter : { mult : true/false }*
 - roots = liste des Id de départ (sommet de l'arbre de classement)

HTTP POST/PUT
/ressources

```
{
  roots : [ liste Id ],
  queries : [
    { query1 },
    { query2 }
  ],
  filter : { filter },
  data : { data }
}
```

HTTP GET /ressources

```
{
  roots : [ liste Id ],
  queries : [
    { query1 },
    { query2 }
  ],
  filter : { filter },
  projection : { projection }
}
```

HTTP PATCH /ressources

```
{
  roots : [ liste Id ],
  queries : [
    { query1 },
    { query2 }
  ],
  filter : { filter },
  action : { action }
}
```

HTTP DELETE /ressources

```
{
  roots : [ liste Id ],
  queries : [
    { query1 },
    { query2 }
  ],
  filter : { filter }
}
```

Domain Specific Language Vitam Query

- Une Query est exprimée avec des opérateurs (inspiré MongoDB / Elasticsearch)

Catégorie	Opérateur	Arguments	Commentaire
Accès direct	<code>\$path</code>	identifiants	Accès direct à un nœud
Booléens	<code>\$and</code> , <code>\$or</code> , <code>\$not</code>	opérateurs	Combinaison logique d'opérateurs
Comparaison	<code>\$eq</code> , <code>\$ne</code> , <code>\$lt</code> , <code>\$lte</code> , <code>\$gt</code> , <code>\$gte</code>	Champ et valeur	Comparaison de la valeur d'un champ et la valeur passée en argument (nombre, date)
	<code>\$range</code>	Champ, <code>\$lt</code> , <code>\$lte</code> , <code>\$gt</code> , <code>\$gte</code> et valeurs	Comparaison de la valeur d'un champ avec l'intervalle passé en argument (nombre, date)
Existence	<code>\$exists</code> , <code>\$missing</code> , <code>\$isNull</code>	Champ	Existence d'un champ
Tableau	<code>\$in</code> , <code>\$nin</code>	Champ et valeurs	Présence de valeurs dans un tableau
	<code>\$size</code>	Champ et taille	Taille d'un tableau
	<code>[n]</code>	Position ($n \geq 0$)	Élément d'un tableau
Textuel	<code>\$term</code> , <code>\$wildcard</code>	Champ, mot clef	Comparaison de champs mots-clefs
	<code>\$match</code> , <code>\$matchPhrase</code> , <code>\$matchPhrasePrefix</code>	Champ, phrase, <code>\$max_expansions</code>	Recherche de phrase
	<code>\$regex</code>	Champ, Expression	Recherche via une expression régulière
	<code>\$search</code>	Champ, valeur	Recherche du type moteur de recherche
	<code>\$flt</code> , <code>\$mlt</code>	Champ, valeur	Recherche « More Like This »
Géomatique	<code>\$geometry</code> , <code>\$box</code> , <code>\$polygon</code> , <code>\$center</code>	Positions	Définition d'une position géographique
	<code>\$geoWithin</code> , <code>\$geoIntersects</code> , <code>\$near</code>	Une forme	Recherche par rapport à une forme géométrique

Domain Specific Language Vitam

Modèle général de réponse à une Query

- **Format de réponse**
 - **Hits** : donne les informations de positionnement de la réponse
 - **Query** : rappelle les informations de requêtes
 - **Results** : donne les résultats (partiels)
- **Format d'erreur**
 - **Context** : contexte d'exécution de l'erreur
 - **Code** : code HTTP, **State** : code court
 - **Message** : message court
 - **Description** : message long
 - **Errors** : erreurs sous-jacentes liées

```
{
  "code": 999, "context": "xxx", "state": "xxx",
  "message": "string",
  "description": "string",
  "errors": [
    { "code": 401, "context": "c1", "state": "s1", "message": "s1", "description": "long s1" },
    { "code": 402, "context": "c2", "state": "s1", "message": "s2", "description": "long s2" },
    { "code": 403, "context": "c3", "state": "s1", "message": "s3", "description": "long s3" }
  ]
}
```

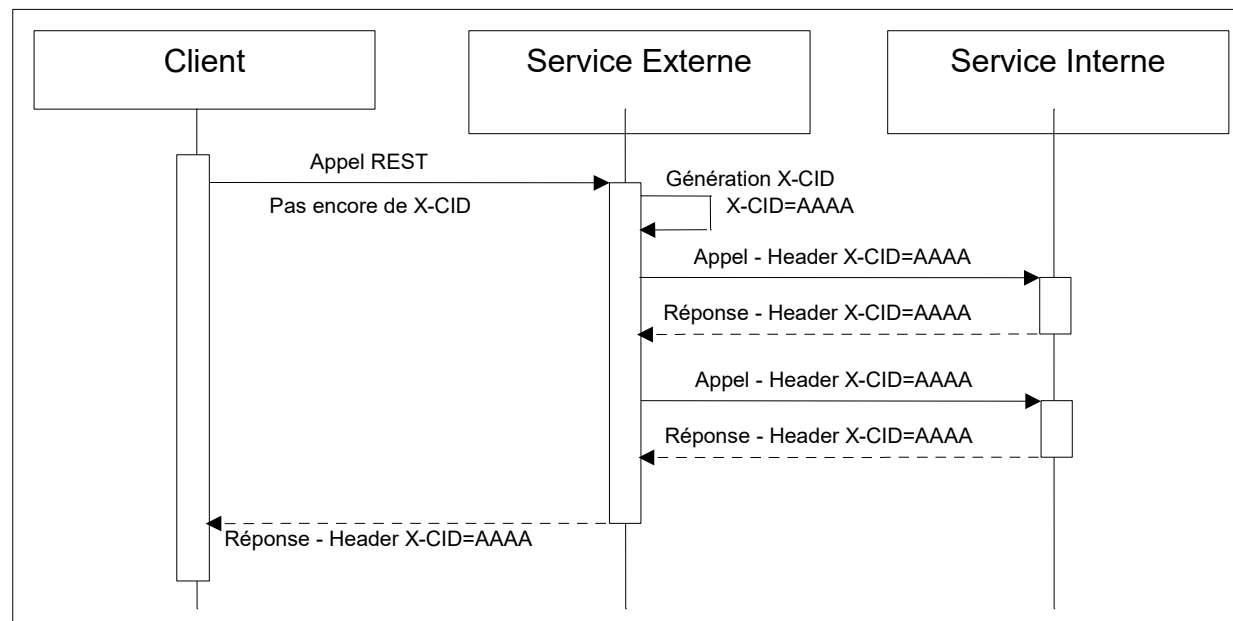
```
{
  hits : {
    total : 67,
    offset : 12,
    limit : 50,
  },
  query : {
    roots : [ liste d'Id ],
    queries : [
      { query1 },
      { query2 }
    ],
    filter : { filter },
    projection : { projection }
  },
  "results" : [
    { ... },
    { ... },
    ...
  ]
}
```

- Usuellement en HTML
 - `<link rel='self' href='https://xxxxx/collections/123456'/>`
- Format Json
 - Pour chaque Result du tableau « results »
 - `'_links': [{ 'rel': 'self', 'href': 'https://xxxxx/collections/123456' }]`
 - Limiter à quelques cas « pratiques » et significatifs
 - *Dossiers parents (Units cf. plus loin)*
 - `'rel' : 'parent'`
 - *Existence de documents sous-jacents réunis dans un ObjectGroup*
 - `'rel' : 'objectgroup'`
 - Mais pas à tous les cas possibles
 - *Dossiers fils*
 - *Potentiellement trop nombreux*
 - *Tous les documents d'un Groupe d'Objets*
 - *Dépend des formats attendus*
 - *Mais étude pour faciliter l'accès à des usages*
 - » `'object_conservation', 'object_hd', 'object_ld', 'object_thumbnail'`

Domain Specific Language Vitam

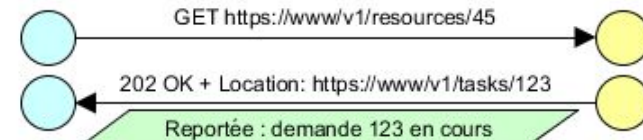
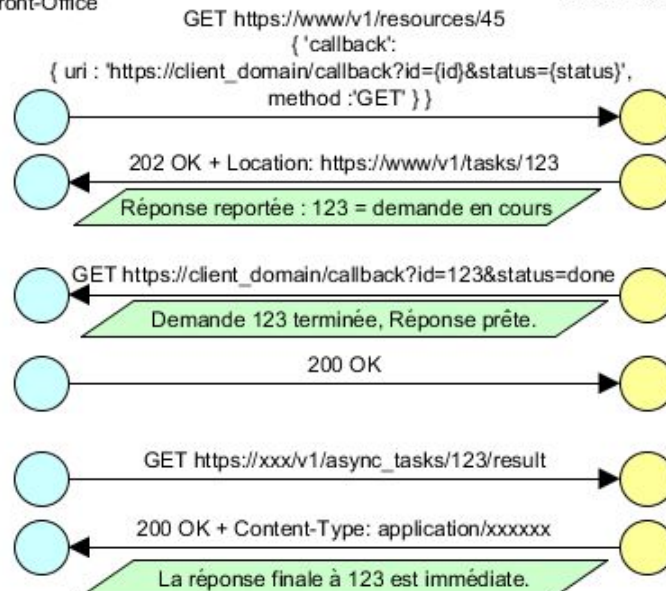
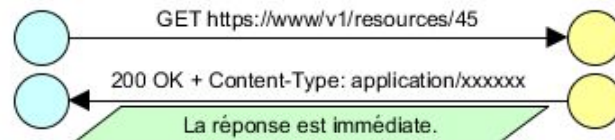
Modèle général de réponse

- Identifiant de corrélation : « **X-CID** »
 - Permet de tracer les événements de part et d'autre (Front-office et Back-office), l'identifiant étant généré par Vitam
 - Permet si nécessaire faire le lien entre une action (vu du Back-office) et un utilisateur (vu du Front-office)
 - Il sera possible d'intégrer un identifiant non signifiant (**X-AID**) en provenance de l'application Front-office pour journalisation dans Vitam avec la transaction
 - *Permet la factorisation des informations dans l'application Front-office sur des notions de session utilisateur (X-AID = Id de session par exemple)*

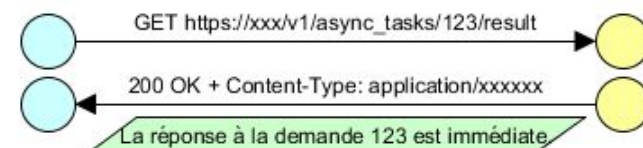
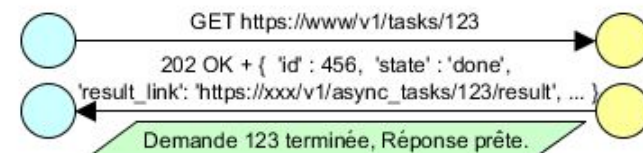
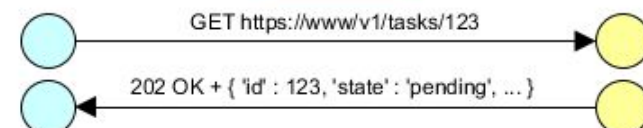


- Modèles approfondis d'implémentation
 - Modèle général
 - Modèles d'API par usage
 - *Authentication et Multi-tenants*
 - *Modèle de requêtes : Domain Specific Language*
 - *Tâches asynchrones*
 - *Modèle de données*
 - *Accès*
 - *Versement*
 - *Gestion*
 - Outillage
 - Exemples (annexe)

Tâches asynchrones



...



Tâches asynchrones

- Ressource : `/async_tasks/`
 - **id** : identifiant technique de la tâche ;
 - **type** : type de tâche ;
 - **state** : statut de la tâche : 'pending', 'done', 'error' ;
 - **start_date** : date de soumission de la tâche ;
 - **end_date** : date de passage de l'état 'pending' à l'état 'end' ou 'error', 'null' tant que la tâche est à l'état 'pending' ;
 - **expiration_date** : la date d'expiration est fixée lors du passage de la ressource à l'état 'done' ou 'error', elle pourra être fixée selon les besoins fonctionnels.
 - *Exemple : end_date+24h. La valeur de ce champ reste à 'null' tant que la tâche est à l'état 'pending'. Lorsque la date d'expiration est dépassée, un mécanisme de purge peut alors libérer les ressources ;*
 - **result_link** : lien vers la sous-ressource permettant l'accès au résultat, 'null' tant que la tâche est à l'état 'pending' ;
 - **error** : la valeur de ce champ reste à 'null' tant que la tâche n'est pas à l'état 'error'. Le format est identique à celui d'un retour en body d'erreur.
- Un GET sur cette ressource retourne un 202 tant que la tâche est à l'état 'pending'.

Tâches asynchrones

- Exemple Pooling : GET https://xxx/v1/async_tasks/456

```
202 OK
{
  'id' : 456,
  'state' : 'pending',
  'type' : 'snapshot'
  'start_date' : '2014-01-10T03:06:17.396Z',
  'end_date':null,
  'expiration_date':null,
  'result_link':null,
  'error':null
}
```

Pooling par le client

```
200 OK
{
  'id' : 456,
  'state' : 'done',
  'type' : 'snapshot'
  'start_date' : '2014-01-10T03:06:17.396Z',
  'end_date':'2014-01-20T03:06:17.396Z',
  'expiration_date':'2014-02-20T03:06:17.396Z',
  'result_link': 'https://xxx/v1/async_tasks/456/result',
  'error':null
}
```

Tâches asynchrones

- Callback :

```
GET https://xxx/v1/servers/5/snapshot
```

```
< {  
<  'callback': {  
<    uri : 'https://client_domain/callback?id={id}&status={status}',  
<    method : 'GET'  
<  }  
< }
```

Réponse :

202 OK

Location: https://xxx/v1/async_tasks/456

Une fois la tâche asynchrone terminée

Callback par le serveur :

GET https://client_domain/callback?id=456&status=done

- Modèles approfondis d'implémentation
 - Modèle général
 - Modèles d'API par usage
 - *Authentication et Multi-tenants*
 - *Modèle de requêtes : Domain Specific Language*
 - *Tâches asynchrones*
 - *Modèle de données*
 - *Accès*
 - *Versement*
 - *Gestion*
 - Outillage
 - Exemples (annexe)

Units

- Description d'archives : métadonnées de description et archivistiques (gestion)
- Un dossier (plan de classement) et/ou une description d'un item.
- Porte l'arborescence d'un plan de classement, éventuellement multiples :
 - *Plusieurs racines (Units de plus haut niveau)*
 - *Plusieurs parents (un sous-dossier peut être rattaché à plusieurs dossiers)*
- SEDA : ArchiveUnit, Isad(G) / EAD : Description Unit
- Au plus un seul groupe d'objets d'archives est attachée à une Unit.
- Un même groupe d'objets d'archives peut être attaché à de multiples Unit.

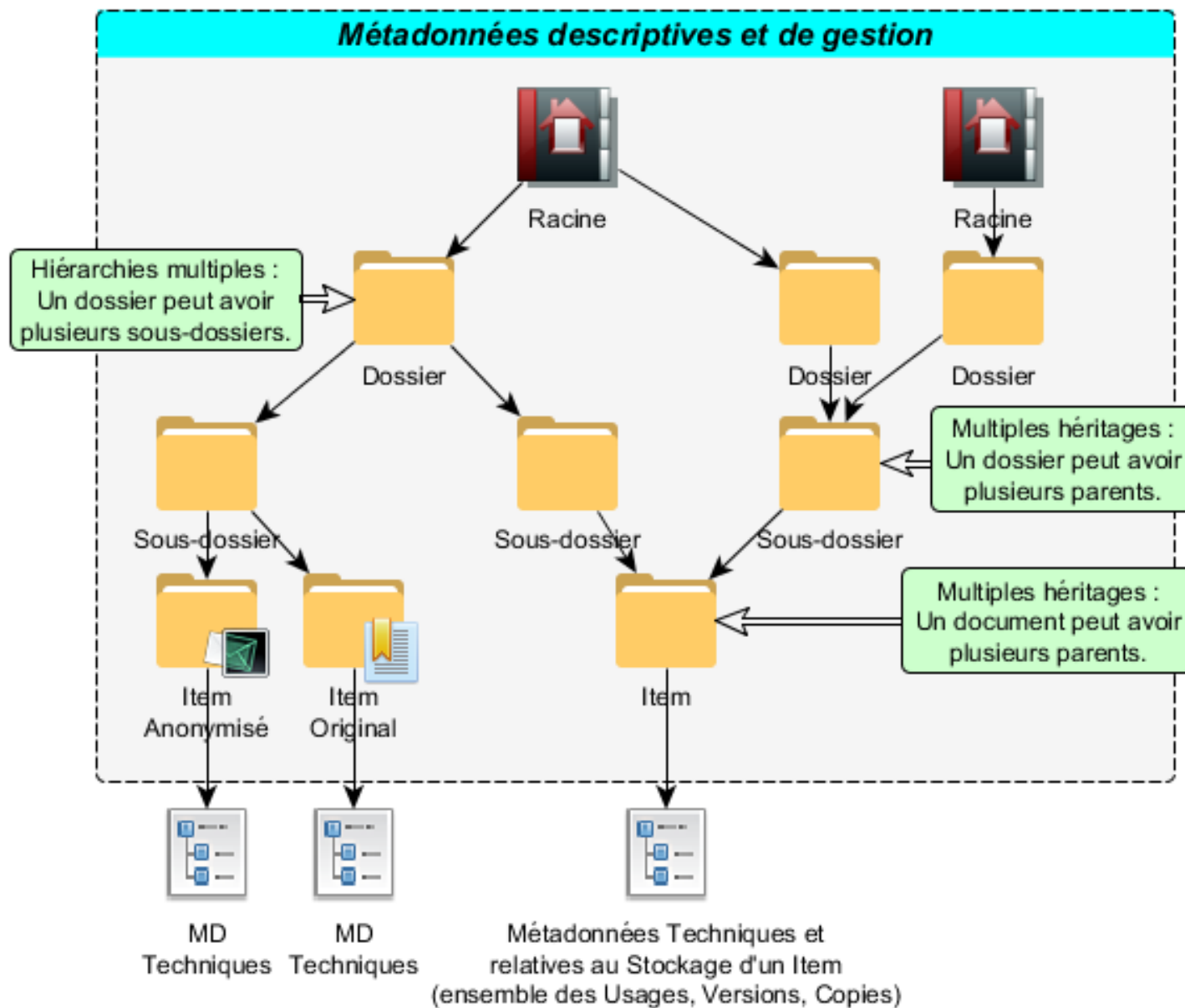
ObjectGroups

- Objets d'archives : binaires ou non binaires (référence à des objet d'archives physiques ou externes au système) et métadonnées techniques
- Plusieurs versions (Objets) pour différencier des usages comme version de conservation, version de diffusion...
- SEDA : DataObjectGroup, EAD : Digital Archive Object Group
- Chaque Groupe d'Objets est attaché à au moins un parent Unit

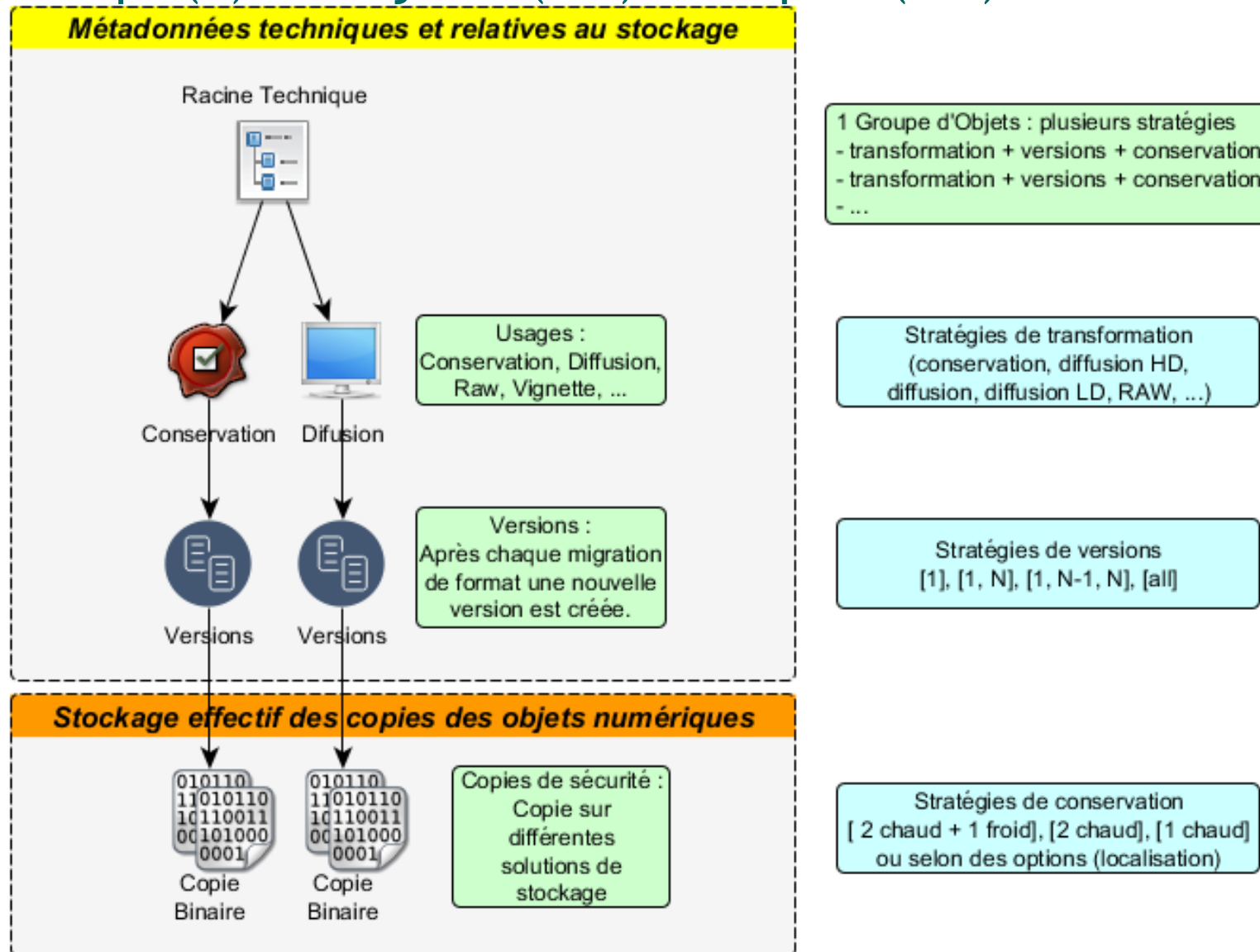
Objects

- Fichiers : par usage et version d'archives dans un Groupe et métadonnées techniques
- SEDA : DataObject (BinaryDataObject ou PhysicalDataObject), EAD : Digital Archive Object

- Units



- ObjectGroups (1) et Objects (~ 2) et Copies (~ 4)



- Modèles approfondis d'implémentation
 - Modèle général
 - Modèles d'API par usage
 - *Authentication et Multi-tenants*
 - *Modèle de requêtes : Domain Specific Language*
 - *Tâches asynchrones*
 - *Modèle de données*
 - Accès
 - *Versement*
 - *Gestion*
 - Outillage
 - Exemples (annexe)

- Accès aux Unités

- La requête utilise le langage de requête (DSL) de Vitam en entrée
- Retourne un JSON format décrit avant avec en résultat une liste d'identifiants uniques en cas de succès et/ou des objets JSON contenant les métadonnées demandées

/units GET (select) PATCH (mise à jour massive)
/units/{id_unit} GET (select) PATCH (mise à jour)
/units/{id_unit}/check GET (valeur probante) ou End Point précédent et argument
/units/{id_unit}/object_groups GET (select)
/units/{id_unit}/object_groups/check GET (valeur probante) ou End Point précédent et argument
/units/{id_unit}/object_groups/{id_object_group} GET (select)
/units/{id_unit}/object_groups/{id_object_group}/check GET (valeur probante)
/units/{id_unit}/object_groups/{id_object_group}/objects GET (select)
/units/{id_unit}/object_groups/{id_object_group}/objects/check GET (valeur probante)
/units/{id_unit}/object_groups/{id_object_group}/objects/{id_object} GET (select)
/units/{id_unit}/object_groups/{id_object_group}/objects/{id_object}/check GET (valeur probante)

- Accès aux Groupes d'Objets et aux Objets

- La requête utilise le langage de requête (DSL) de Vitam en entrée
- Retourne un JSON format décrit avant avec en résultat une liste d'identifiants uniques en cas de succès et/ou des objets JSON contenant les métadonnées demandées ou le binaire selon l'option
 - *accept : application/json, application/octet-stream*
- /object_groups GET (select)
- /object_groups/{id_object_group} GET (select)
- /object_groups/{id_object_group}/check GET (valeur probante) ou End Point précédent et argument
- /object_groups/{id_object_group}/objects GET (select)
- /object_groups/{id_object_group}/objects/check GET (valeur probante)
- /object_groups/{id_object_group}/objects/{id_object} GET (select)
- /object_groups/{id_object_group}/objects/{id_object}/check GET (valeur probante)
- /objects GET (select)
- /objects/{id_object} GET (select)
- /objects/{id_object}/check GET (valeur probante)

- Modèles approfondis d'implémentation
 - Modèle général
 - Modèles d'API par usage
 - *Authentication et Multi-tenants*
 - *Modèle de requêtes : Domain Specific Language*
 - *Tâches asynchrones*
 - *Modèle de données*
 - *Accès*
 - *Versement*
 - *Gestion*
 - Outillage
 - Exemples (annexe)

- L'API de Versement (Ingest) 1 / 2
 - La version la plus simple est celle utilisant le modèle SEDA
 - */ingests* GET (select) POST (creation)
 - *La commande POST contient*
 - *Soit un ZIP contenant tout*
 - *Soit un multipart avec le fichier XML SEDA et les fichiers numériques (Objects) associés*
 - *La requête est asynchrone*
 - *La commande GET permet de lister les versements en cours*
 - *La commande POST est similaire à un transfert de fichier via un autre protocole*
 - */ingests/{id_async}* GET (select status = tâche asynchrone) DELETE (annulation)

• L'API de Versement (Ingest) 2/2

- Permet une version sous forme d'une transaction de versement
 - *Adapté à des versements fréquents de petites tailles*
- **/ingests/** est le point d'entrée pour le processus de versement
 - *Il reproduit le SEDA mais sous forme « constructive », bien adapté pour de petits versements*
 - *Il suit la logique d'accès à « units » et « object_groups » mais selon les modes de création, mise à jour et effacement*
 - *Les opérations sur des Units ou ObjectGroups pré-existants sont autorisées uniquement en mode mise à jour (Units) et ajout d'Objets (ObjectGroups).*
 - *Toutes les autres opérations s'appliquent uniquement aux Units, ObjectGroups et Objects nouvellement créés.*
- **/ingests** GET (select) POST (création)
- **/ingests/{id_async}** GET (select) PATCH (finalisation avant exécution) DELETE (annulation)
- **/ingests/{id_async}/units** GET (select) POST (création) PATCH (mise à jour) DELETE (effacement)
- **/ingests/{id_async}/units/{id_unit}** GET (select) PATCH (mise à jour) DELETE (effacement)
- ...
- **/ingests/{id_async}/object_groups** GET (select) POST (création) DELETE (effacement)
- **/ingests/{id_async}/object_groups/{id_object_group}** GET (select) PUT (ajout) DELETE (effacement)
- ...

- Modèles approfondis d'implémentation
 - Modèle général
 - Modèles d'API par usage
 - *Authentication et Multi-tenants*
 - *Modèle de requêtes : Domain Specific Language*
 - *Tâches asynchrones*
 - *Modèle de données*
 - *Accès*
 - *Versement*
 - *Gestion*
 - Outillage
 - Exemples (annexe)

Exemple : Opérations de destruction

- Les listes des Units et ObjectGroups à éliminer est produite par les batch de préparation de destruction
 - *La sélection se fait sur la base des métadonnées de gestion relatives à l'élimination*
 - *La mise à jour sur la liste produite porte sur ces mêmes métadonnées de gestion et permet de retirer en conséquence des archives du lot d'élimination*
 - *La validation de la liste (PATCH sur `/destructions/{id_sync}`) permet de lancer le batch de finalisation qui vérifie à nouveau les métadonnées de gestion avant de procéder aux destructions*
- Une transaction de destruction est une tâche asynchrone volatile, ce qui signifie qu'elle peut disparaître dès qu'elle est terminée. Cependant les Journaux (logbooks) garderont une trace de celle-ci.
- `/destructions` GET (select)
- `/destructions/{id_async}` GET (select) PATCH (finalisation) DELETE (annulation)
- `/destructions/{id_async}/units` GET (select) PATCH (mise à jour de masse)
- `/destructions/{id_async}/units/{id_unit}` GET (select) PATCH (mise à jour)
- `/destructions/{id_async}/units/{id_unit}/object_groups` GET (select)
- `/destructions/{id_async}/units/{id_unit}/object_groups/{id_object_group}` GET (select)
- ...

- Exemple : Accès aux journaux

- Les journaux sont de 3 natures
 - *Journal du SAE : journal central rassemblant tous les événements*
 - Sur les archives, les grands éléments du journal des opérations et des journaux du cycle de vie
 - Sur le SAE, les éléments liés aux évolutions du SAE (mise à jour, activation / désactivation / modification de configurations, arrêt/relance, ...)
 - */logbooks GET (select)*
 - */logbooks/{id_day} GET (select)*
 - *Journal des opérations : journal regroupant les entrées, les éliminations, les transformations, ... par lot d'archives (filiales)*
 - */operation_logbooks GET (select)*
 - */operation_logbooks/{id_op} GET (select)*
 - *Journal du cycle de vie : journal par Unit et par ObjectGroup retraçant toute la vie de l'item*
 - */units/{id_unit}/lifecyle GET (select)*
 - */object_groups/{id_object_group}/lifecyle GET (select)*

- Modèles approfondis d'implémentation
 - Modèle général
 - Modèles d'API par usage
 - *Authentication et Multi-tenants*
 - *Modèle de requêtes : Domain Specific Language*
 - *Tâches asynchrones*
 - *Modèle de données*
 - *Accès*
 - *Versement*
 - *Gestion*
 - Outillage
 - Exemples (annexe)

- Documentation
 - Quelle forme ?
- Exemple de codes
 - Quelle forme ?
- Framework de génération de requêtes et de parsing de réponses
 - Permettrait de générer un « body » pour une requête HTTPS vers un Vitam
 - Permettrait de parser un « body » de réponse d'un Vitam
- Framework de connexions (protocole HTTPS)
 - Va plus loin que la génération ou le parsing des contenus
 - Gère les URI et les connexions HTTPS
 - *Pooling*
 - *Chiffrement, authentication*
 - *Mais limité à des configurations « standards » (authentication à base de certificat par exemple)*
 - *Limité à une bibliothèque de support réseau potentiellement divergente de votre framework applicatif, rendant difficile la gestion en mode flux des réponses*
 - » *Limite les possibilités de « pass-through » pour les réponses binaires (contenu d'une archive)*

Outillage et Intégration

- Quelles seraient les limites à l'intégration de Vitam dans votre SI ?
 - Il s'agit d'identifier des coûts récurrents qui pourraient être mutualisés.
 - *Sur un plan Application Front-Office ?*
 - *Sur un plan Intégration dans votre SI (outillage de votre production, règles, sécurité, ...) ?*
 - *Sur un plan Exploitation dans votre SI (organisations, compétences, formations, ...) ?*
 - Que pourriez vous attendre du Programme Vitam ?

- Modèles approfondis d'implémentation
 - Modèle général
 - Modèles d'API par usage
 - *Authentication et Multi-tenants*
 - *Modèle de requêtes : Domain Specific Language*
 - *Tâches asynchrones*
 - *Modèle de données*
 - *Accès*
 - *Versement*
 - *Gestion*
 - Outillage
 - Exemples (annexe)

Exemple d'usage

Versement d'un lot d'archives

- Supposition : Fichier SEDA et ZIP contenant tous les fichiers plus le fichier SEDA

1) POST <https://vitam/ingest/v1/ingests>

Header : authentication + id contrat de versement

Body : ZIP

2) Réponse 201 (requête créée)

Header : X-CID (id transaction), id_ingest

3) GET https://vitam/ingest/v1/ingests/id_ingest

Header : X-CID, authentication

4) Réponse 202 (requête toujours en cours)

Header : X-CID (id transaction), id_ingest

Répétition des étapes 3 et 4 : pooling

5) GET https://vitam/ingest/v1/ingests/id_ingest

Header : X-CID, authentication

6) Réponse 200 (requête terminée) ou 40x (erreur)

Header : X-CID (id transaction), id_ingest

Body : Résultat JSON (détail du versement ou de l'erreur)

Exemple d'usage

Versement d'un lot d'archives : variante 2

- Supposition : Fichier SEDA et ZIP contenant tous les fichiers plus le fichier SEDA avec variante Callback
 - 1) **POST** <https://vitam/ingest/v1/ingests>
Header : authentication + id contrat de versement + callback : uri
Body : ZIP
 - 2) **Réponse 201 (requête créée)**
Header : X-CID (id transaction), id_ingest
 - ... temps de traitement
 - 3) **Vitam appelle** https://URI_CALLBACK?X_CID=id&ID=id_ingest&status=ok/ko
Header : X-CID (id transaction), id_ingest
 - 4) **GET** https://vitam/ingest/v1/ingests/id_ingest
Header : X-CID, authentication
 - 5) **Réponse 200 (requête terminée) ou 40x (erreur)**
Header : X-CID (id transaction), id_ingest
Body : Résultat JSON (détail du versement ou de l'erreur)

Exemple d'usage

Versement d'un lot d'archives : variante 3 1/3

- Supposition : Métadonnées pour chaque dossier et chaque fichier avec une arborescence mais pas de SEDA
 - *MD globales*
 - */Dossier1 + MD1*
 - */Dossier1/Dossier1.1 + MD1.1*
 - */Dossier1/Dossier1.1/Fichier1.1.1 + MD 1.1.1*
 - */Dossier1/Dossier1.1/Fichier1.1.2 + MD 1.1.2*
- 1) **POST** <https://vitam/ingest/v1/ingests>
 - Header : authentication + id contrat de versement*
 - Body : MD globales*
- 2) **Réponse 201 (requête créée)**
 - Header : X-CID (id transaction), id_ingest*
- Création d'un ObjectGroup
- 3) **POST** https://vitam/ingest/v1/ingests/id_ingest/object_groups
 - Header : X-CID, authentication*
 - Body : Multipart MD1.1.1 + Fichier1.1.1*
- 4) **Réponse 200 (groupe d'objet créé) et identifiant og111**

Exemple d'usage

Versement d'un lot d'archives : variante 3 2/3

- Suite : les Units et les liens avec les ObjectGroups
 - 5) POST https://vitam/ingest/v1/ingests/id_ingest/units
Header : authentication + id contrat de versement
Body : MD1
 - 6) Réponse 200 (Unit créée) et identifiant unit1
Header : X-CID (id transaction), id_ingest
 - 7) POST https://vitam/ingest/v1/ingests/id_ingest/units
Header : X-CID, authentication
Body : MD1.1 + liaison avec unit1
 - 8) Réponse 200 (Unit créée) et identifiant unit11
- Lien avec un ObjectGroup déjà créé
 - 9) PATCH https://vitam/ingest/v1/ingests/id_ingest/units/unit11/object_groups/og111
Header : X-CID, authentication
 - 10) Réponse 201 (groupe d'objet lié)
- Lien avec un ObjectGroup nouvellement créé
 - 11) POST https://vitam/ingest/v1/ingests/id_ingest/units/unit11/object_groups
Header : X-CID, authentication
Body : Multipart MD1.1.2 + Fichier1.1.2
 - 12) Réponse 200 (groupe d'objet créé et lié) et identifiant og112

Exemple d'usage

Versement d'un lot d'archives : variante 3 3/3

- Suite : finalisation

15) PATCH https://vitam/ingest/v1/ingests/id_ingest

Header : authentication + id contrat de versement

Body : status=ready

16) Réponse 201 (requête asynchrone acceptée)

Header : X-CID (id transaction), id_ingest

17) GET https://vitam/ingest/v1/ingests/id_ingest

Header : X-CID, authentication

18) Réponse 202 (requête toujours en cours)

Header : X-CID (id transaction), id_ingest

19) GET https://vitam/ingest/v1/ingests/id_ingest

Header : X-CID, authentication

20) Réponse 200 (requête terminée) ou 40x (erreur)

Header : X-CID (id transaction), id_ingest

Body : Résultat JSON (détail du versement ou de l'erreur)

Exemple d'usage

Versement d'un lot d'archives : variante 4

- Supposition : Métadonnées pour chaque dossier et chaque fichier avec une arborescence mais pas de SEDA et 2 versions d'un même groupe d'objets
 - */Dossier1/Groupe1.1/version papier + MD 1.1.1*
 - */Dossier1/Groupe1.1/version scan Fichier1.1.2 + MD 1.1.2*
 - *Zoom sur le changement ObjectGroups (aucun changement sur le reste)*
- 1) POST https://vitam/ingest/v1/ingests/id_ingest/object_groups
Header : X-CID, authentication
Body : Multipart MD1.1.1 + information version papier
 - 2) Réponse 201 (groupe d'objets créé) et identifiant groupe d'objets og111 et objet o111
 - 3) POST https://vitam/ingest/v1/ingests/id_ingest/object_groups/og111/objects
Header : X-CID, authentication
Body : Multipart MD1.1.2 + Fichier1.1.2 + information version scan
 - 4) Réponse 201 (objet créé et ajouté au groupe d'objets) et identifiant o112

Exemple d'usage

Élimination d'archives

- **Supposition : Un lot d'archives (identifié par 1 filière) est candidat à être éliminé**
 - 1) **GET** <https://vitam/management/v1/destructions>
Header : authentication + id contrat d'accès, Body : query { queries : [{ \$eq : { filière : « filièreId » } }] }
 - 2) **Réponse 200**
Header : X-CID (id transaction), Body : { hits : ..., query : ..., results : [{ identifiant de la liste id_async }] }
 - 3) **GET** https://vitam/management/v1/destructions/id_async
Header : X-CID, authentication
 - Permet d'afficher le lot d'archives, par types, dates, contenus, ...
 - 4) **Réponse 200 + Body = lot d'archives**
 - 5) **PATCH** https://vitam/management/v1/destructions/id_async/units
Header : X-CID, authentication
 - Mise à jour massive d'un sous-ensemble (query) de Units sur les métadonnées de gestion
 - 6) **Réponse 200 + Body = Résultat de l'opération**
 - 7) **PATCH** https://vitam/management/v1/destructions/id_async
Header : X-CID, authentication, Body : status = ready
 - Permet de valider le lot de destruction (que Vitam va réanalyser suite aux modifications)
 - 8) **Réponse 201 (requête créée)**
 - 9) **GET** https://vitam/management/v1/destructions/id_async
Header : X-CID, authentication
 - 10) **Réponse 200 (requête terminée) ou 40x (erreur)**
Header : X-CID (id transaction), id_async, Body : Résultat JSON
 - Permet de récupérer les informations de destructions effectives

Exemple d'usage

Accès au journal des entrées, des éliminations

- **Supposition : connaissance de l'identifiant d'une filière**

- 1) **GET** https://vitam/management/v1/operation_logbooks

Header : authentication + id contrat d'accès, Body : filtre sur type d'opérations (entrée / élimination) et sur la filière concernée

- 2) **Réponse 200**

Header : X-CID (id transaction)

Body : { hits : ..., query : ..., results : [{ liste de id_async }] }

- Permet d'afficher la liste des opérations (entrée, élimination, ...) selon un filtre

- 3) **GET** https://vitam/management/v1/operation_logbooks/id_async

Header : X-CID, authentication

- Permet d'afficher le contenu du journal concerné (accès direct possible)

- 4) **Réponse 200**

Body = contenu du journal (de l'entrée ou de l'élimination)

- 5) **GET** https://vitam/management/v1/operation_logbooks

Header : X-CID, authentication

Body : indiquant le contenu attendu (détail de l'entrée, statut de chacune des Units de l'entrée, notamment en termes d'élimination, ...) pour une filière donnée

- Permet la construction d'un journal d'élimination par filière ou par entrée

- 6) **Réponse 200**

Body = contenu du journal demandé (déduit des journaux du cycle de vie le cas échéant)

Exemple d'usage

Recherche d'une unité d'archives

- **Supposition : Métadonnées à rechercher est « title = mon titre »**
 - 1) **GET** <https://vitam/access/v1/units>
Header : authentication + id contrat d'accès
Body : query { queries : [{ \$eq : { title : « mon titre » } }] }
 - 2) **Réponse 200**
Header : X-CID (id transaction)
Body : { hits : { total : 1, offset : 0, limit : 50 },
query { queries : [{ \$eq : { title : « mon titre » } }] },
results : [{ toutes les métadonnées du Unit + référence à l'ObjectGroup refog }] }
 - 3) **GET** https://vitam/access/v1/object_groups/refog
Header : X-CID, authentication, X-Format:vignette
accept : application/octet-stream
 - Permet d'afficher les métadonnées et la vignette quasi simultanément dans l'IHM
 - 4) **Réponse 200 + Body = Usage Vignette, Dernière version**
 - 5) **GET** https://vitam/access/v1/object_groups/refog
Header : X-CID, authentication, X-Format:diffusion
accept : application/octet-stream
 - L'utilisateur a cliqué sur la vignette pour voir le document en grand
 - 6) **Réponse 200 + Body = Usage Diffusion, Dernière version**

Exemple d'usage

Recherches complexes

- **Supposition :**

- Recherche 1 : chercher les dossiers « nom=Dupont » tel que un sous-dossier est « activité=fin »
- Recherche 2 : chercher les dossiers « propriété=x » tel que l'objet binaire associé soit de type « Video »

- 1) GET https://vitam/access/v1/units

Header : authentication + id contrat d'accès

Body : query { queries : [

{ \$eq : { nom : « Dupont » } },

{ \$eq : { activité : « fin » }, \$depth : 1 }, // \$depth => profondeur (ici fils immédiat) (défaut=)*

{ \$depth : -1 }] } // \$depth => profondeur inverse (ici parent immédiat)

- **Réponse 200**

Header : X-CID (id transaction)

Body : { hits : { total : 10, offset : 0, limit : 50 }, query : ...,

results : [liste des Units répondant aux critères] }

- 2) GET https://vitam/access/v1/units

Header : authentication + id contrat d'accès

Body : query { queries : [

{ \$eq : { propriété : « x » } },

{ \$eq : { type : « VIDEO » }, \$objectgroup : 1 }, // \$objectgroup => profondeur de groupe d'objets

{ \$unit : -1 }] } // \$unit => profondeur inverse (ici Unit parent immédiat)

- **Réponse 200 similaire**

- **Il est possible de faire la même chose en multiples requêtes, moins efficace**

1) GET /Units { \$eq : { nom : « Dupont » } }

2) GET /Units roots : [liste résultat précédent] { \$eq : { activité : « fin » }, \$depth : 1 }

3) GET /Units roots : [liste résultat précédent] { \$depth : -1 }