# BEYOND - Data Factory - Dossier d’architecture

## Préambule

Ce document est une version issue d’une documentation en ligne.

La version à jour de ce document est disponible à l’adresse suivante :

[Documentation en ligne](https://bitbucket.org/sxdibm/byd-all-documentation/src/master/01.byd-DataFactory/1.ArchitectureDossier/0000.Introduction.md)

Ce document est géré en configuration comme le serait du code.

Cela veut dire que la version la plus à jour est celle publiée en ligne dans la branche master et que la mise à jour de ce document est continue.

Si un export (en pdf ou au format Microsoft Word) reste possible, il est conseillé de consulter ce document au travers de son interface WEB.

Dans le cadre d’une lecture d’un fichier exporté, des effets de bord pourraient apparaître. Par exemple, la résolution de liens internes permettant la navigation dans le document en ligne.

Les contributeurs (Auteurs, relecteurs) pourront amener leurs modifications directement dans le document dans des ‘branches’. Les mécanismes de “Merge” permettront aux administrateurs de la documentation de les prendre en compte.

Alternativement, une fiche “classique” de relecture est fournie à l’adresse suivante. Le relecteur veillera à bien intégrer l’identifiant de ‘commit’ (disponible sur chaque page) permettant de référencer la version en relecture.

Une traçabilité des changements faits et de leurs auteurs sur le document est conservée et est consultable en ligne.

## Introduction

Ce document est le dossier d’architecture de la plateforme **BEYOND - Data Factory**

Dans sa version électronique, le lecteur pourra pointer directement vers un des chapitres au moyen de la table des matières.

## Chapitres

Les chapitres de ce document couvriront les thèmes suivants :

* [Introduction](./0000.Introduction.md)
* [Table des matières](./0001.TableOfContent.markdown)
* [Glossaire](./0002.Glossary.md)
* [Contexte du Système](./0100.SystemContext.md)
* [Modélisation des cas d’utilisation](./0200.UseCaseModel.md)
* [Exigences](./0300.Requirements.md)
* [Architecture : vue d’ensemble](./0400.ArchitectureOverview.md)
* [Modélisation des composants](./0500.ComponentsModel.md)
* [Décisions d’architecture](./0600.ArchitectureDecisions.md)
* [Modèle de déploiement](./0700.DeploymentModel.md)
* [Modèle de données](./0800.DataModel.md)
* [Liste des logiciels](./0900.COTSList.md)

## Document de référence

|  |  |  |
| --- | --- | --- |
| Document | Nom de fichier | Emplacement |
| . |  |  |

## Auteurs

|  |  |
| --- | --- |
| Rôle | NOM - mail |
| Auteurs | Vincent ROBINET / Sylvain WILBERT |
| Valideurs |  |
| Approbateurs | . |

# Glossaire

## Acronymes

La liste des acronymes utilisés dans le dossier d’architecture est :

|  |  |
| --- | --- |
| ACRONYMES | DEFINITION |
| **API** | Application Programming Interface |
| **REST** | Representational State Transfer |
| **SGBD** | Système de Gestion de Base de Données |
| **BDD** | Base De Données |
| **SIG** | Système d’information géographique |
| **BIM** | Building Information Modeling |
| **GMAO** | Gestion de maintenance assistée par ordinateur |
| **IdP** | Identity Provider |
| **KPI** | Key Performance Indicator - Indicateur “clé” de performance |
| **COTS** | Commercial Off The Shelf: Solution logicielle disponible sur le ‘marché’ |
| **NoSQL** | Not Only SQL : famille de systèmes de gestion de base de données qui s’écarte des stratégies des bases relationnelles |
| **LB** | Load Blancer |

## Termes

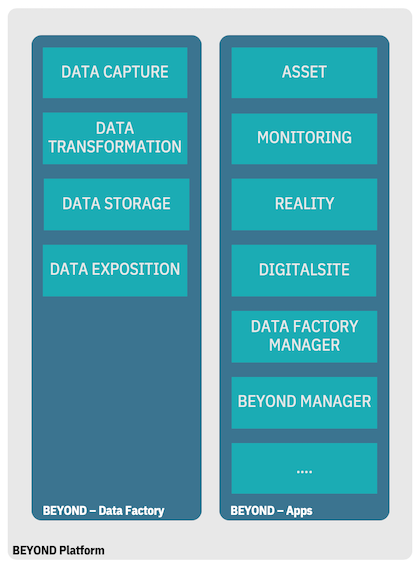
Les termes suivants seront utilisés dans le dossier d’architecture :

|  |  |
| --- | --- |
| **TERMES** | **DESCRIPTION** |
| **Microservice** | Une architecture par **microservices** est un style d’architecture logicielle où la complexité apparente des applications est diminuée par la décomposition des processus en plusieurs entités indépendantes et faiblement couplées. Ils sont souvent exposés au travers d’API REST (reposant sur du JSON/Http(s)). |
| **“Tenant”** / **Entité** | Un des “clients” du SYSTEME. Ses processus, ses données sont isolées des autres ‘tenants’ tout en partageant la même plateforme |
| **“Multi-tenancy”** | Capacité de la plateforme à pouvoir accueillir et offrir ses services applicatifs à plusieurs tenants différents en préservant la confidentialité, l’intégrité des échanges et des données avec chacun d’eux |
| **Produit** | Dans le contexte de BEYOND, un **produit** désigne un service informatique *généralisable* qui répond à un besoin identifié des entités utilisatrices (ou tenants). Cette plateforme permet après configuration et contextualisation d’intégrer les données des tenants (Accostage par API) et d’adapter les fonctions proposées dans la limite du périmètre prévu par la plateforme (Pas de développement spécifique aux différents « tenants »). |

| **Load Balancer**| un load balancer a pour tâche de répartir la charge de travail sur plusieurs serveurs (serveur Web / mail / BDD) |

# Diagramme de contexte du système

## Périmètre



Contexte du Système

L’écosystème **BEYOND** est une plateforme de données et de services dans le monde de la construction. A ce titre, elle peut être segmentée en 2 grands modules :

* **BEYOND Data Factory** : Ce module permet l’acquisition, le stockage, la transformation, et l’exposition des données à des tiers
* **BEYOND Apps** : Ce module permet d’exposer des services applicatifs basés sur les données de la Data Factory.

Le module **BEYOND Data Factory** est l’objet de ce dossier d’architecture. Toutefois, il faut noter qu’une application **BEYOND Data Factory Manager** permet de gérer la **BEYOND Data Factory**. Cette application, comme tous les services applicatifs de la plateforme reposera sur les principes architecturaux de **BEYOND Apps**. Néanmoins, pour des raisons de lisibilité et afin de traiter le sujet “Data Factory” dans son ensemble, elle sera aussi decrite dans ce document.

Le module **BEYOND Data Factory** et l’application de gestion **BEYOND Data Factory Manager** seront dénommés **SYSTEME** dans le reste du document.

## Introduction

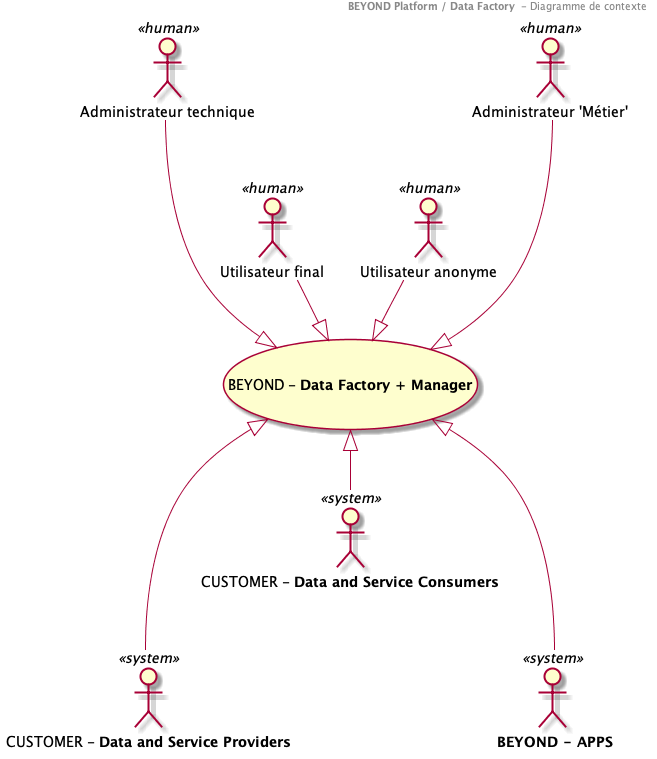
Le diagramme de contexte du système est une représentation de l’architecture de la solution à son plus haut niveau.

Il permet de représenter

* Les frontières du **SYSTEME**,
* Son rôle et ses responsabilités
* Les interactions auxquelles il est soumis
  + Par les utilisateurs (Fonctions dans l’application)
  + Par les systèmes externes.

A ce stade, le diagramme de contexte du SYSTEME ne détaille pas la nature (technologique, fonctionnelle) de ces interactions qui seront décrites par les cas d’utilisation et la vision générale de l’architecture.

## **BEYOND Data Factory** / **BEYOND Data Factory Manager** a.k.a. le SYSTEME



Contexte du Système

### Rôles et responsabilités

Le **SYSTEME** est une plateforme visant à :

* **CAPTURER** la donnée auprès de tiers externes.
* **STOCKER** la donnée sous un mode permettant la traçabilité et l’auditabilité des mouvements.
* **TRANSFORMER** la donnée pour la normaliser, l’enrichir, la rendre ‘opérable’.
* **EXPOSER** la donnée pour la rendre accessible aux utilisateurs (BEYOND Apps, ou consommateurs externes).

Aussi les responsabilités du **SYSTEME** sont de :

* Fournir des canaux d’acquisition de données versatiles (Synchrone, Asynchrone).
* Contrôler que la donnée fournie est saine (anti-virus).
* Assurer l’intégrité de la donnée transmise en s’assurant que le fichier reçu est conforme à celui émis.
* Valider l’éligibilité (technique et fonctionnelle) d’une donnée entrante.
* Permettre le stockage de la donnée en l’association à un “tenant”, un propriétaire.
* Identifier de manière unique la donnée reçue.
* Permettre l’annotation d’une donnée.
* Appliquer des algorithmes de transformation à la donnée.
* Assurer la traçabilité des mouvements des données (Écriture, Consultation).
* Fournir une capacité de recherche de la donnée.
* Exposer la donnée à des tiers.

### Frontières

### Interactions

Le **SYSTEME** interagit avec les utilisateurs suivants :

**Fonction d’administration** :

* **Administrateur technique** : en charge du paramétrage technique de la solution. Il est en charge de suivre les éléments de journalisation et notamment les alertes afin d’y remédier.
* **Administrateur ‘Métier’** : en charge du suivi des métriques “métier” remontées par le SYSTEME. Il est en charge de suivre les éléments de journalisation “Métier” et notamment les alertes afin d’y remédier. Il est aussi en charge du paramétrage permettant de contextualiser le fonctionnement de la plateforme. Par exemple, il définit les profils de permissions pouvant être associés aux utilisateurs de la plateforme.

**Utilisateurs standard** :

* **Utilisateurs finaux** : Utilisateur authentifie de la plateforme. Le **SYSTEME** repose sur des profils d’utilisateurs permettant de composer les permissions atomiques disponibles. Une permission permet d’ajouter une fonction élémentaire.
* **Utilisateurs anonymes** : Avant tout action d’authentification un utilisateur peut utiliser une URL BEYOND, à minima une page d’atterrissage doit être positionnée. Par ailleurs, des parcours utilisateurs d’incitation seront disponibles à terme sur la plateforme en mode anonyme. Le **SYSTEME** interagit avec les systèmes suivants

…

# Diagrammes de cas d’utilisation

## Acteurs

Les différents acteurs peuvent être répartis selon 2 catégories :

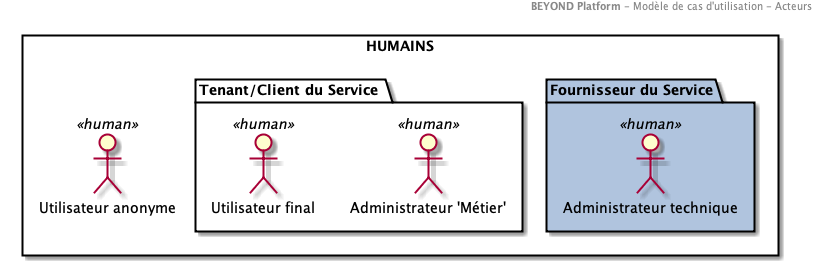
* **Acteurs “solutions” :** services, systèmes ou applications
* **Acteurs humains :** utilisateurs finaux de la plateforme

Les acteurs “humains” sont catégorisés en différentes populations, chacune avec des rôles et des droits spécifiques sur la plateforme.

Remarque : La notion d’acteur permet de décrire l’intervenant “Métier” dans son utilisation du **SYSTEME**.

On parle de **profil** ‘utilisateur’ pouvant agréger un ensemble de **permissions** atomiques. Certaines permissions peuvent alors être partagées par différents profils.

### Acteurs Humains



Acteurs ‘humains’

#### Administrateur technique (HUMAIN) - Niveau “Produit”

**Rôle** : Assure le paramétrage du SYSTEME pour les paramètres non accessibles des “Tenants”. **Responsabilités** :

* **Configure** les paramètres du système (Par exemple eligibilité des fichiers entrants)
* **Consulte** les journaux techniques de la plateforme

#### Administrateur ‘Métier’ (HUMAIN) - Niveau “Tenant”

**Rôle** : Assure le paramétrage du SYSTEME pour personnaliser l’espace dédié aux “Tenants”. **Responsabilités** :

#### Utilisateur anonyme (HUMAIN)- Niveau “Tenant” color:red (ne voit pas à quoi ça correspond ? )

**Rôle** : Utilisateur non authentifié accédant le **SYSTEME**

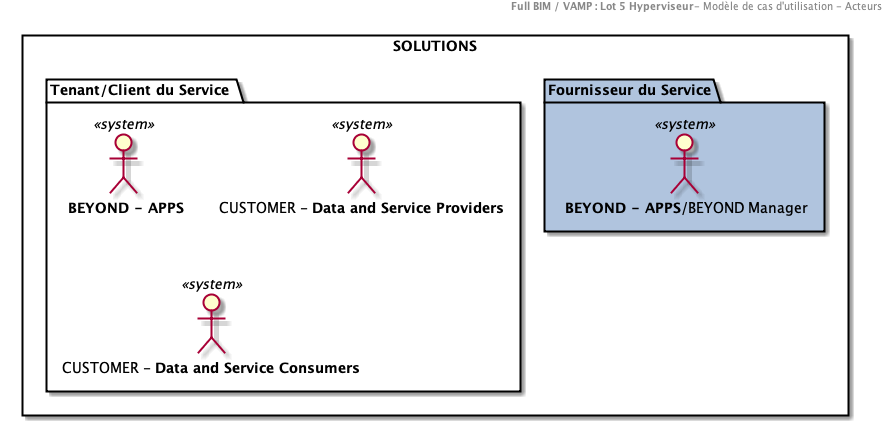
**Responsabilités** : N/A

#### Utilisateur final (HUMAIN)- Niveau “Tenant”

**Rôle** : Utilisateur authentifié accédant le **SYSTEME**

**Responsabilités** : N/A

### Acteurs “solutions”



Acteurs ‘solutions’

#### BEYOND - APPS / BEYOND Manager

**Rôle** : BEYOND Manager est une application offrant l’ensemble des services transverses de la plateforme aux BEYOND Apps.

**Responsabilités** :

* Service d’authentification des utilisateurs.
* Service d’identification de haut niveau des utilisateurs.
* Service de tracabilité : Collecte des évènements de la plateforme.

#### BEYOND - APPS

**Rôle** : Les BEYOND Apps sont les premières consommatrices des données du **SYSTEME**. Par exemple, **BEYOND Reality** est une consommatrice des données 3D exposées par le **SYSTEME**

**Responsabilités** :

#### CUSTOMER - Data and Service Providers

**Rôle** : Fournisseurs de données externes au **SYSTEME**

**Responsabilités** :

#### CUSTOMER - Data and Service Consumers

**Rôle** : Consommateurs de données externes au **SYSTEME**

**Responsabilités** :

## En tant qu’utilisateur final (authentifié) autorisé, je peux remonter un fichier brut en vue de son traitement par le SYSTEME

### Introduction

Ce cas d’utilisation permet la remontée en toute sécurité d’un fichier dans l’écosystème afin de l’inscrire dans sa file de traitement.

Le processus suivant est envisagé :

1. Validation des droits de l’utilisateur
2. Récupération des caractéristiques de la remontée du fichier
3. Sélection du fichier depuis un poste local
4. Contrôle du type de fichier
5. Contrôle de la taille du fichier
6. Vérification de l’existence du fichier dans le corpus BEYOND
7. Remontée du fichier
8. Indexation du fichier dans BEYOND (annotations, champs additionnels)
9. Inscription du fichier dans la file d’attente de traitement correspondante

Remarques Ce cas d’utilisation envisage une remontée manuelle par un utilisateur depuis une interface graphique. Une alternative de remontée automatisé en traitement M2M est aussi disponible (cf. cas d’utilisation correspondant) Des cas d’utilisation supplémentaire son définit pour l’implémentation du traitement)

### Acteurs

Un utilisateur final authentifié auprès de de **BEYOND MANAGER**

### Préconditions

1 - L’utilisateur final dispose du profil “Utilisateur” suivant tel que défini dans **BEYOND Apps - BEYOND Manager** , par l’administrateur ‘Métier’ (Niveau BEYOND MANAGER):

* **[DATA-UPLOAD]**: Profil permettant d’afficher la fonction de remontée de fichiers manuelle depuis le **SYSTEME**
* **[DATA-TYPE001]** : Profil permettant la remontée de fichier de type 001. (la configuration du type de fichier s’opérant dans l’application DataFactoryManager - paramétrage local)

L’administrateur ‘Métier’ (Niveau **SYSTEME**) a configuré localement le type **[DATA-TYPE001]** afin

* de définir le type de fichier (par ex : .las) basé sur l’extension de celui ci. La vérification par le processus se fera en revanche sur le type MIME dans le cas ou celui est disponible.
* de définir une taille maximale (par ex : 1Go)
* un traitement associé (optionnel) parmi les types de traitement disponibles (Par exemple : “CESIUM-Transformation”)

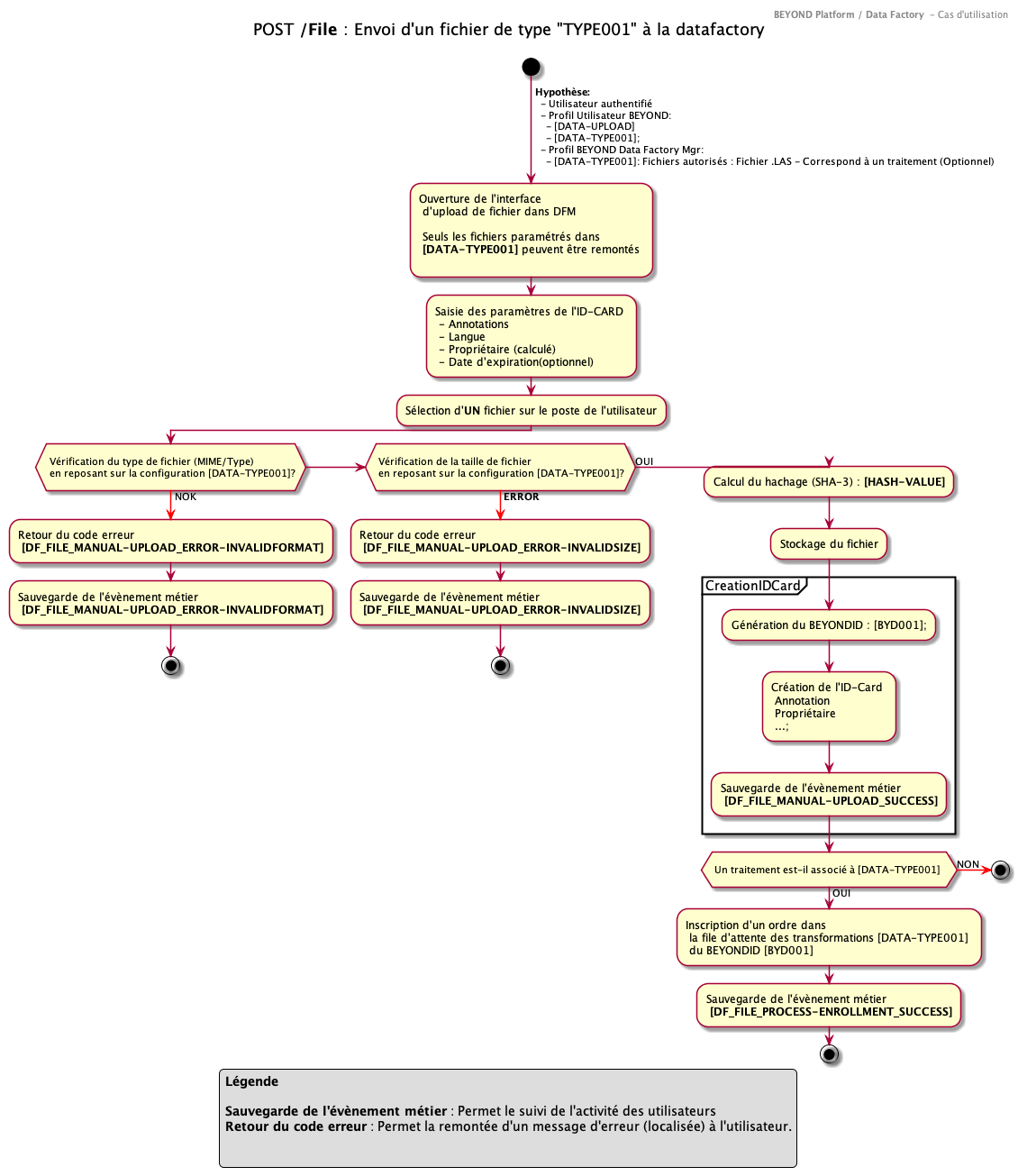
**Remarque** la cohérence des valeurs est une responsabilité des administrateurs.

### Déclencheur

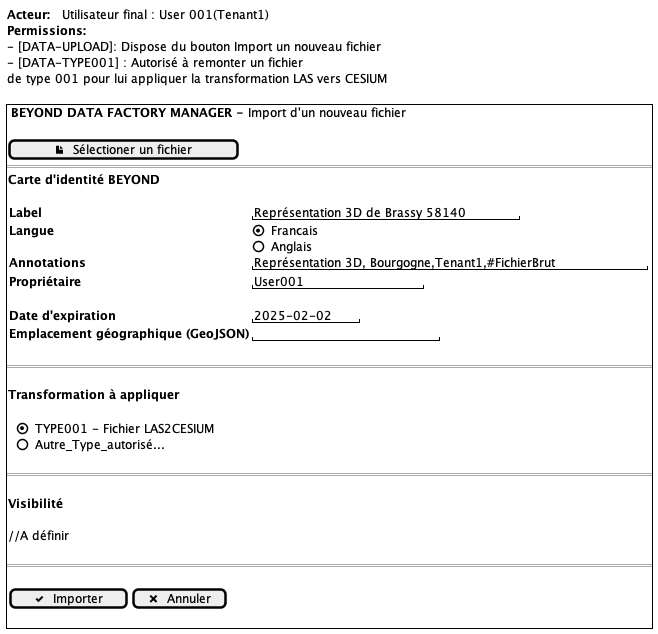
L’utilisateur accède à BEYOND, s’authentifie puis accède à la page de remontée de fichier du **SYSTEME** (Data Factory Manager)

### Description

Remarque : La gestion des antivirus, sont considérés comme une capacité technologique fournie par défaut sur les “Cloud Object Storage”et ne nécéssitent pas un déploiement spécifique. A ce titre, il ne figure pas comme un traitement à lancer par le **SYSTEME** dans le cas d’utilisation.



Cas d’utilisation

Exemple d’écrans 

### Alternatives

* Cas d’utilisation : Remontée de fichiers M2M

## Lancement d’un traitement sur un fichier brut reçu par le SYSTEME

### Introduction

Ce cas d’utilisation permet l’application d’un traitement dont l’ordre d’exécution vient d’être posté dans la file d’attente.

Le processus suivant est envisagé :

1. Chaque traitement dispose de sa propre file d’attente. (Un fichier devant subir 2 traitements, aura donc 2 ordres de traitement dans 2 files d’attente différentes et les traitements seront exécutés independemment et en fonction de son rang dans la file d’attente)
2. Récupération de la configuration du traitement à exécuter
3. Récupération du fichier à traiter au moyen de son BYDID renseigné dans l’ordre d’exécution
4. Application du traitement sur le fichier brut
5. En cas de succès, création d’un nouvel objet BEYOND (Nouvel BYDID2)
6. Le processus est suivi par une journalisation d’évènements métier qui agit comme un composant de ‘Business Activity Monitoring’.

### Acteurs

Processus du **SYSTEME**

### Préconditions

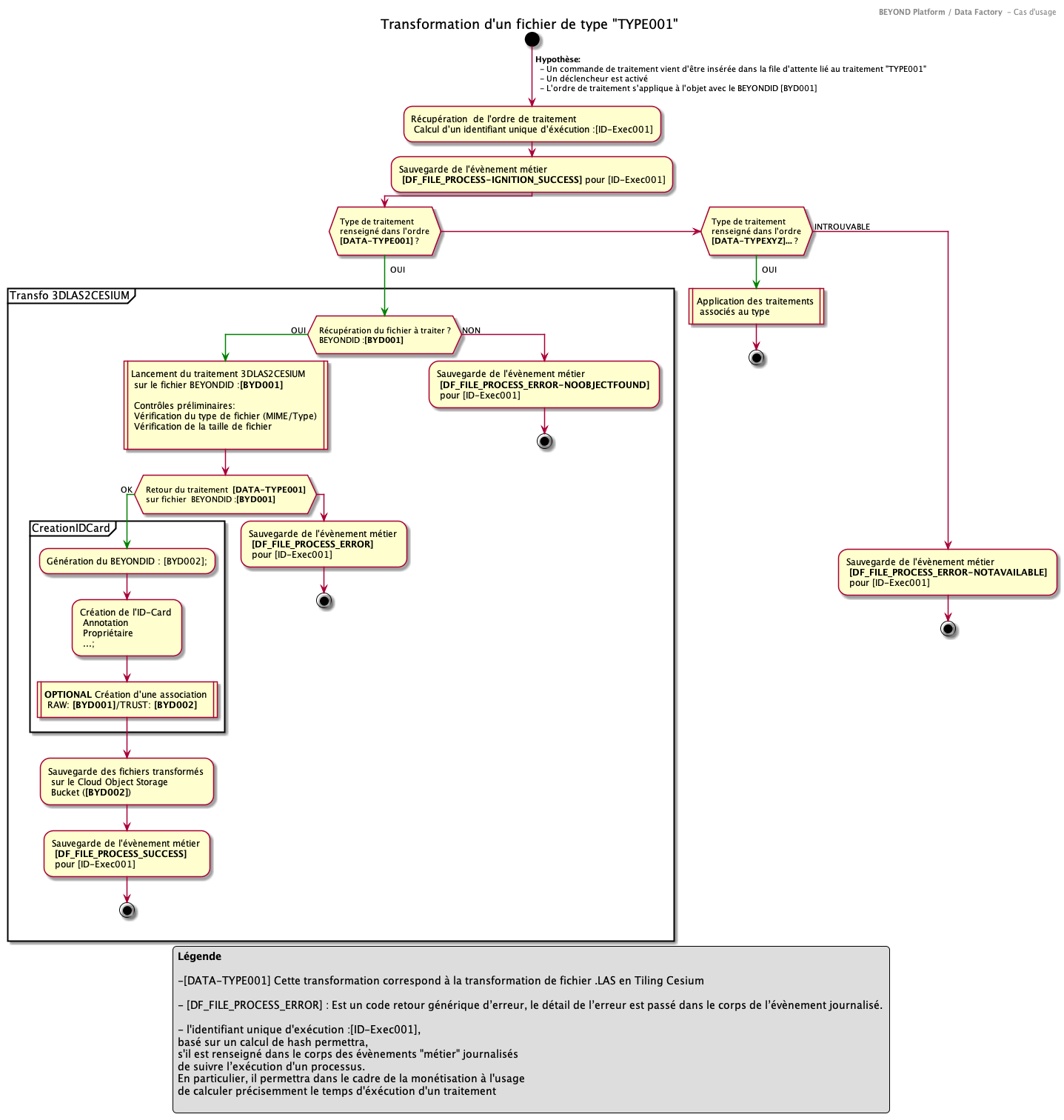
1. Le fichier brut existe et porte le BYDID : **BYD001**
2. Le traitement des fichier Las-> CESIUM TIling est implémenté. Il est configuré dans le **SYSTEME** (depuis la console d’administration de la Data Factory Manager) pour être associé au l’identifiant de traitement **[DATA-TYPE001]**

### Déclencheur

Le processus est déclenché par l’arrivé d’un ordre d’exécution dans la file de traitement.

### Description

**Remarque** Le lancement d’un traitement imposera l’exécution des contrôles sur le type de fichiers et la taille. Si cela peut paraitre redondant (car déjà fait lors de la phase d’UPLOAD), cela permettra de garantir la cohérence du processus pour les fichiers en file d’attente dont le paramétrage du traitement aurait changé depuis le chargement (approche par couplage faible).



Cas d’utilisation

### Alternatives

N/A

# Exigences

## Introduction

Une exigence permet de caractériser le SYSTEME par ses propriétés telles que sa performance, sa robustesse, sa convivialité, sa maintenabilité, les fonctions qu’il couvre, ou la manière d’intégrer des tiers externes.

Les exigences seront catégorisées par niveau de criticité

* **OBLIGATOIRE** : Fonction nécessaire à la définition du produit minimal viable. (MUST HAVE)
* **ESSENTIEL** : Fonction nécessaire à la définition du produit à la cible , mais pas forcément indispensable dans les premières versions. (SHOULD HAVE)
* **OPTIONNEL** : Fonction qui n’est pas vitale à l’utilisation de la solution. (COULD HAVE)

Les exigences décrites dans ce document visent à suivre les préceptes “SMART”

* **Specific/Spécifique** : Chaque exigence doit être précise pour éviter toute place à l’interprétation.
* **Measurable/Mesurable** : Chaque exigence doit pouvoir être mesurée afin d’en valider l’atteinte.
* **Attainable/Accessible** : Chaque exigence doit être accessible (réalisable) selon des circonstances actuelles données.
* **Realisable/Réaliste** : Chaque exigence doit être considérée comme réaliste (représentative d’une réalité connue ou cible).
* **Traceable/Traçable** : Chaque exigence doit être reliée à son implémentation. Notamment une matrice de couverture permettant d’associer, la solution , à ses exigences (techniques et fonctionnelles) devra être maintenue.
* [Exigences non fonctionnelles](./0301.NonFunctionalRequirements.md)
* [Exigences fonctionnelles](./0302.FunctionalRequirements.md)
* [Exigences d’intégration](./0303.IntegrationRequirements.md)

# Exigences non fonctionnelles

Remarque : [Fichier du détail des exigences non fonctionnelles](./tools/03.Requirements/LOT5-ArchitectureDossier-REQUIREMENTS.xlsx)

## Introduction

Les exigences non fonctionnelles d’une solution peuvent se catégoriser entre

* Ses **qualités intrinsèques** de la solution.
* Les **contraintes extérieures** qui s’appliquent à elle.

A- Qualités

Les qualités d’un SYSTEME se déclinent entre

1. les qualités présentées lors de **‘l’exécution’** de la solution :
   * Accessibilité
   * Capacité
   * Disponibilité
   * Fiabilité
   * Intégrité
   * Manageabilité
   * Performance
   * Sécurité
   * Utilisabilité
2. les qualités intégrées lors de **‘la conception’** de la solution :
   * Portabilité
   * Reprise
   * Scalabilité
   * Maintenabilité

B- Contraintes

Les contraintes s’appliquant à un SYSTEME se déclinent entre

* Les contraintes techniques :
  + Emplacement géographique
  + Intégration du système d’information de SIXENSE
  + Internationalisation
  + Normes et Standard
* Les contraintes “Métier” :
  + Directives du Groupe SIXENSE
  + Législation en vigueur

## Qualités

### Accessibilité

Pas d’exigences exprimées.

### Capacité

|  |  |  |  |
| --- | --- | --- | --- |
| **Identifiant** | Description | Origine | Criticité |
| **BYD-DF\_NFR\_017** | Le *SYSTEME* doit : accueillir YYYY utilisateurs **potentiels** | Hypothèse | OBLIGATOIRE |
| **BYD-DF\_NFR\_018** | Le *SYSTEME* doit : accueillir des utilisateurs **courants** (Connectés au système, mais pas forcément actifs) à hauteur de 10% des utilisateurs potentiels | Hypothèse | ESSENTIEL |
| **BYD-DF\_NFR\_019** | Le *SYSTEME* doit : accueillir des utilisateurs **actifs/concurrents** à hauteur de 1% des utilisateurs potentiels | Hypothèse | ESSENTIEL |
| **BYD-DF\_NFR\_020** | Le *SYSTEME* doit : pouvoir répondre à **plusieurs sollicitations** (API) concurrentes par des systèmes externes | Hypothèse | OBLIGATOIRE |

#### Sollicitations des utilisateurs

Pas d’exigences exprimées.

#### Sollicitations des systemes tiers

Pas d’exigences exprimées.

### Disponibilité

|  |  |  |  |
| --- | --- | --- | --- |
| **Identifiant** | Description | Origine | Criticité |
| **BYD-DF\_NFR\_024** | Le *SYSTEME* doit : proposer une plage de service remontant une disponibilité correspondant a un SLA de 99.5% en dehors des interruptions pour maintenance programmée | Hypothèse | OBLIGATOIRE |
| **BYD-DF\_NFR\_025** | Le *SYSTEME* doit : programmer des plages de maintenance d’une durée de 4h par mois | Hypothèse | ESSENTIEL |

### Fiabilité

Pas d’exigences exprimées.

### Intégrité

|  |  |  |  |
| --- | --- | --- | --- |
| **Identifiant** | Description | Origine | Criticité |
| **BYD-DF\_NFR\_007** | Le *SYSTEME* doit : permettre la ségrégation des données par client (“Multi-tenant”) | Hypothèse | OBLIGATOIRE |
| **BYD-DF\_NFR\_011** | Le *SYSTEME* doit : stocker toutes les informations de date/temps selon la zone **UTC Zero** | Hypothèse | OBLIGATOIRE |
| **BYD-DF\_NFR\_013** | Le *SYSTEME* doit : stocker les informations de date/temps au format **ISO 8601** Exemple : **2017-09-15T17:27:00Z** | Hypothèse | OBLIGATOIRE color:red (certainement meilleur en timestamp GMT) **Réponse SWT**: UTC Zulu/Zero me semble bien meilleur/ universel : cf. article “UTC et GMT L’utilisation de l’appellation standard temps moyen de Greenwich (sigle : GMT, de l’anglais Greenwich Mean Time) s’était imposée par la prépondérance de la marine britannique durant le xixe siècle. Elle est désormais déconseillée parce que sa définition est ambiguë. L’utilisation de la nouvelle appellation normalisée temps universel coordonné (ou son abréviation UTC) doit lui être préférée.” -> Quelle est la position de la DT ? |
| **BYD-DF\_NFR\_015** | Le *SYSTEME* doit : stocker les informations de codes pays au format **ISO 3166** Exemple : **fr** | Hypothèse | OBLIGATOIRE |
| **BYD-DF\_NFR\_016** | Le *SYSTEME* doit : stocker les informations d’unités monétaires au format **ISO 4217** Exemple : **EUR** | Hypothèse | OBLIGATOIRE |
| **BYD-DF\_NFR\_029** | Le *SYSTEME* doit : mettre en place des mécanismes permettant d’optimiser la qualité des données entrantes (Contrôle d’intégrité, Liste de valeurs finies) | Hypothèse | ESSENTIEL |
| **BYD-DF\_NFR\_030** | Le *SYSTEME* doit : journaliser les évènements ‘métier’ (opérations sur les données) à des fins de traçabilité | Hypothèse | OBLIGATOIRE |
| **BYD-DF\_NFR\_035** | Le *SYSTEME* doit : journaliser les évènements techniques | Hypothèse | OBLIGATOIRE |

### “Manageabilité”

|  |  |  |  |
| --- | --- | --- | --- |
| **Identifiant** | Description | Origine | Criticité |
| **BYD-DF\_NFR\_001** | Le *SYSTEME* doit : doit implémenter une chaine d’intégration et de déploiement continue (DEVOPS - CI/CD)afin de permettre un temps de mise sur le marché (TTM) optimisé. | Hypothèse | ESSENTIEL |
| **BYD-DF\_NFR\_031** | Le *SYSTEME* doit : promouvoir le couplage faible entre composants (internes et externes) | Hypothèse | OBLIGATOIRE |

### Performance

|  |  |  |  |
| --- | --- | --- | --- |
| **Identifiant** | Description | Origine | Criticité |
| **BYD-DF\_NFR\_026** | Le *SYSTEME* doit : authentifier un utilisateur anonyme en 30 secondes | Hypothèse | ESSENTIEL |
| **BYD-DF\_NFR\_027** | Le *SYSTEME* doit : afficher 90% des pages Web en moins de 3 secondes | Hypothèse | ESSENTIEL |
| **BYD-DF\_NFR\_028** | Le *SYSTEME* doit : afficher les pages web en moins de 15 secondes | Hypothèse | ESSENTIEL |
| **BYD-DF\_NFR\_032** | Le *SYSTEME* doit : absorber un pic de charge correspondant à **25%** de la charge totale quotidienne concentrée sur un pic d’une heure. | Hypothèse | ESSENTIEL |

### Sécurité

|  |  |  |  |
| --- | --- | --- | --- |
| **Identifiant** | Description | Origine | Criticité |
| **BYD-DF\_NFR\_021** | Le *SYSTEME* doit : reposer sur un référentiel des identités fourni par un tiers externe (OpenID Connect, SAML, …) pour l’ensemble de ses utilisateurs (Interne/Externe) (Authentification: Identification de haut niveau : Accès à l’application) | Hypothèse | OBLIGATOIRE |
| **BYD-DF\_NFR\_022** | Le *SYSTEME* doit : reposer sur un gestionnaire des identités interne au SYSTEME permettant d’associer des permissions à un utilisateur | Hypothèse | OBLIGATOIRE |
| **BYD-DF\_NFR\_023** | Le *SYSTEME* doit : reposer sur le protocole https (Combinaison entre HTTP et un protocole de chiffrement) pour tous les échanges : (client serveur, serveur serveur). | Hypothèse | OBLIGATOIRE |
| **BYD-DF\_NFR\_036** | Le *SYSTEME* doit : reposer sur un référentiel des identités fourni par Sixense Digital pour l’ensemble des utilisateurs d’aministration du produit (Authentification: Identification de haut niveau : Accès à l’application) | Hypothèse | OBLIGATOIRE |

### Utilisabilité

|  |  |  |  |
| --- | --- | --- | --- |
| **Identifiant** | Description | Origine | Criticité |
| **BYD-DF\_NFR\_002** | Le *SYSTEME* doit : permettre l’affichage de ses écrans WEB avec une résolution minimale de 1024\*768 color:red (on peut partir plus grand, je ne suis pas que le format existe encore) \*\*SWT :PI :c’est la valeur minimale supportée (permetttant l’ouverture d’un bug). Quelle est la valeur souhaitée ? | Hypothèse | ESSENTIEL |
| **BYD-DF\_NFR\_003** | Le *SYSTEME* doit : permettre l’affichage de ses écrans WEB avec le navigateur **Chrome** Version 75.0. sur Windows 10 (64 Bits). | Hypothèse | OBLIGATOIRE |
| **BYD-DF\_NFR\_004** | Le *SYSTEME* doit : être testé avec un affichage de ses écrans WEB avec le navigateur **Firefox** Version 68 sur Windows 10 (64 Bits). | Hypothèse | ESSENTIEL |
| **BYD-DF\_NFR\_005** | Le *SYSTEME* doit : être testé avec un affichage de ses écrans WEB avec le navigateur **Safari** Version 12 sur MacOs Mojave. | Hypothèse | ESSENTIEL |
| **BYD-DF\_NFR\_006** | Le *SYSTEME* doit : permettre l’affichage de ses écrans WEB dans une approche “responsive” permettant la gestion des orientations ‘Tablette’ | Hypothèse | OBLIGATOIRE |
| **BYD-DF\_NFR\_012** | Le *SYSTEME* doit : Afficher les informations de date/temps dans le fuseau horaire configuré sur le navigateur de l’utilisateur. | Hypothèse | ESSENTIEL |

### Portabilité

|  |  |  |  |
| --- | --- | --- | --- |
| **Identifiant** | Description | Origine | Criticité |
| **BYD-DF\_NFR\_033** | Le *SYSTEME* doit : faciliter la portabilité de la solution vers différents systèmes d’hébergement (Fournisseur Cloud, Sur site) | Hypothèse | OBLIGATOIRE |

### Reprise

Pas d’exigences exprimées.

### Scalabilité

|  |  |  |  |
| --- | --- | --- | --- |
| **Identifiant** | Description | Origine | Criticité |
| **BYD-DF\_NFR\_034** | Le *SYSTEME* doit : permettre d’adapter sa configuration à la charge utilisateurs/systèmes (Croissance/Décroissance) | Hypothèse | OBLIGATOIRE |

### Maintenabilité

Pas d’exigences exprimées.

## Les contraintes

### Intégration du système d’information d’un tiers

Pas d’exigences exprimées.

### Internationalisation

|  |  |  |  |
| --- | --- | --- | --- |
| **Identifiant** | Description | Origine | Criticité |
| **BYD-DF\_NFR\_008** | Le *SYSTEME* doit : afficher l’ensemble des labels de ses interfaces en **français** par défaut | Hypothèse | OBLIGATOIRE |
| **BYD-DF\_NFR\_009** | Le *SYSTEME* doit : permettre l’affichage des labels de ses interfaces dans d’autres langues (Right-To-Left) au moyen du chargement d’un fichier de configuration sur le serveur, selon la configuration du navigateur appelant. | Hypothèse | OPTIONNEL |
| **BYD-DF\_NFR\_010** | Le *SYSTEME* doit : stocker et afficher les éléments journalisés sur la plateforme en **anglais** | Hypothèse | ESSENTIEL |

### Standards

Pas d’exigences exprimées.

### Directives du Groupe SIXENSE

|  |  |  |  |
| --- | --- | --- | --- |
| **Identifiant** | Description | Origine | Criticité |
| **BYD-DF\_NFR\_014** | Le *SYSTEME* doit : doit être localisé en Union Européenne | Hypothèse | OBLIGATOIRE |

### Législation en vigueur

La RGPD sera traitée conjointement avec les équipes Sécurité SIXENSE.

# Exigences fonctionnelles

## Introduction

Les exigences fonctionnelles définissent :

* les processus qu’opèrent le SYSTEME (ce “qu’il fait”)
* les éléments qu’il génère (ce “qu’il produit”)

Ces éléments d’exigences du **SYSTEME** seront décrits dans les cas d’utilisation de la plateforme dont l’ensemble fournira la couverture fonctionnelle à atteindre.

[Modèle des cas d’utilisation](./0200.UseCaseModel.md)

# Exigences sur l’intégration

## Introduction

Les exigences autour de l’intégration des systèmes visent à décrire le mode opératoire (fonctionnel et technique) d’accostage des tiers externes en interaction avec le **SYSTEME**

Pour chacun des intervenants, elles définissent pour chaque fonction :

* Le mode d’accostage physique (Synchrone/Asynchrone/“Bulk Transfer” qui correspond à un transport de type sFTP par exemple)
* La technologie d’accostage sous-jacente
* La description du flux de données vue du **SYSTEME** (émetteur/récepteur)
* La donnée échangée
* La fréquence de requête
* La volumétrie échangée
* Les éléments de sécurité
* Le lien vers le contrat d’interface qui détaille les exigences fonctionnelles (cas passants, exceptions), le contrat technique (wsdl/json).
* Le lien vers les éléments de test
* Bouchons applicatifs
* Données représentatives

## Définition

Définition

Définition

Une interface est un mécanisme d’accostage entre 2 tiers, faisant la promotion d’un couplage faible en formalisant la stratégie d’intégration au travers d’un contrat. Ainsi, la façon dont est produite la donnée exposée est masquée à ceux qui l’utilisent et vice versa.

Pour éviter les ambiguïtés, les définitions suivantes sont proposées

* Le **PRODUCTEUR** est le système qui expose le service aux tiers externes
* Les **CONSOMMATEURS** sont les systèmes consommant la donnée exposée

Une interface entre deux acteurs doit permettre la gestion :

* Du **cas nominal** : Le résultat attendu quand l’interface fonctionne
* Des **cas d’exceptions** : L’ensemble des combinaisons de sortie pour des cas d’erreurs (fonctionnelles et/ou techniques)

Le principe fondateur d’une interface est de mettre en œuvre un système déterministe empêchant toute interprétation (de la requête ou de la réponse). Cela passe par une maitrise complète de cet accostage notamment en le décrivant au travers d’un **contrat d’interface** qui décrit en détail l’accostage technologique entre les tiers, ainsi que la donnée échangée.

Celui-ci peut prendre différents formats en fonction de la technologie en jeu :

* Swagger pour des API Rest (La préconisation pour les échanges avec le **SYSTEME**)
* Wsdl pour des webservices XML on SOAP
* Fichier “doc” pour des accostages type sFTP ou autres.

## Matrice de responsabilité

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Responsible | Accountable | Consulted | Informed |
| Formalisation du **besoin d’interface** | Consommateurs |  |  | Producteur |
| Formalisation du **contrat d’interface** (Document/swagger/wsdl) : Cas nominal, Exceptions | Producteur |  | Consommateurs |  |
| Fourniture des **bouchons applicatifs** avec des données représentatives | Producteur |  |  | Consommateurs |
| Implémentation de la **demi-interface : Exposition du service** | Producteur |  |  | Consommateurs |
| Implémentation de la **demi-interface : Consommation du service** | Consommateurs |  |  | Producteur |
| Test d’intégration de l’interface | Consommateurs |  | Producteur |  |

Remarque: Afin de permettre les implémentations, le **SYSTEME** définira les contrats d’interface dont il est consommateur si ceux ci ne sont pas disponibles (getDetails, getAscendant, getDescedants, getPK, getSiblings)

## Stratégie d’interfaçage

Principe de l’interfaçage

Principe de l’interfaçage

Les enjeux de l’interfaçage du **SYSTEME** (Lot5 - Hyperviseur) avec des tiers externes sont de

* Lui permettre de disposer d’un jeu de données suffisant, contenant les attributs permettant un affichage de l’ensemble des informations informations nécessaires et suffisantes dans les écrans de « synthèse ».
* Lui permettre de récupérer à la volée les données supplémentaires auprès du PRODUCTEUR de la donnée permettant détailler le contexte de l’objet affiché dans les écrans de « détail ».
* Lui permettre de rediriger l’utilisateur du **SYSTEME** vers le PRODUCTEUR de la donnée afin qu’il puisse la modifier, sous reserve que les permissions d’accès en modification lui aient été fournies dans le PRODUCTEUR.
* Conserver la cohérence globale des données qu’il contient, sachant que le **SYSTEME** peut lui aussi être un producteur de données.

La stratégie d’interfaçage est la suivante :

**[1]**: Chacun des lots peut contenir des éléments ( par exemple un actif) qu’il souhaite partager et/ou voir afficher dans le **SYSTEME**. Pour cela il doit être compatible avec les exigences du SYSTÈME, à savoir, chacun des éléments doit être décrit par

* Un jeu d’attributs (“communs”en marron sur le schéma) (Titre, Identifiants, Catégories, Version, emplacement géographique…. – La liste reste à définir à l’aune des conclusions du chantier sur le dictionnaire de données).
* Un jeu d’attributs propres aux fonctions de chaque lot/tiers externe (" spécifiques ", en rose et jaune pales)

Ces attributs sont nécessaires afin

* De garantir un affichage performant et robuste des informations ‘nécessaires et suffisantes’ dans le SYSTÈME.
* De permettre de récupérer dynamiquement les informations détaillant l’élément auprès du PRODUCTEUR de la donnée
* De permettre le débranchement vers le PRODUCTEUR de la donnée afin de pouvoir la modifier, si les droits de l’utilisateur configurés dans le PRODUCTEUR le permettent.

**[2]**: Un déclencheur permet d’envoyer l’élément et son jeu d’attributs communs vers le SYSTÈME (Batch, passage de workflow pour l’élément à l’état « PUBLIC », action utilisateurs…)

**[3]**: L’ensemble des éléments (y compris ceux produits par le **SYSTEME** lui-même) est stocké. A ce stade, le SYSTÈME peut afficher la donnée “nécessaire et suffisante” (autonomie vis à vis des tiers externes)

Différentes options sont possibles pour « lier » les données dans le **SYSTEME**.

* Liens **fournis** : les lots externes renseignent la dépendance
* Liens **déduits** : la structuration des objets permet de les relier (découlant entre autre du dictionnaire des données, du référencement linéaire, du SIG)
* Liens **explicites** : Le **SYSTEME** fournit une interface permettant de lier les objets explicitement

**[4]**: Lors de l’affichage d’un élément dans le SYSTÈME, le détail de celui-ci peut être récupéré dynamiquement auprès du producteur (API REST)

**[5]**: De nouveaux services inédits pourront être générés par le SYSTÈME afin d’exposer de nouveaux objets ‘composites’ reflétant l’accostage des différents lots avec le SYSTÈME.

Remarque : Il est pris en hypothèse que le **SYSTEME** doit tendre à reposer sur/exposer des données **fraiches**, c’est à dire exposant la version la plus ‘à jour’ aux yeux de son propriétaire

### Segmentation des données

Segmentation de la données

Segmentation de la données

### Intérêts de cette approche

**Stratégie “Produit”** : permet de définir un produit générique ou lié à une industrie selon la répartition des attributs Communs/Spécifiques

**Ouverture/Polyvalence** : Permet d’intégrer de nouveaux lots facilement (Respect de la stratégie d’attributs Communs/Spécifiques)

**Couplage Faible** : L’évolution des tiers externes n’impacte pas la structuration du **SYSTEME**

**Performance** : Seules les données essentielles au fonctionnement du **SYSTEME** sont rapatriées.

**Pertinence/fraicheur des données** : Le détail des attributs spécifiques est requeté à la demande

**Innovation** : De nouveaux services inédits sont exposés par le **SYSTEME** (Composition, Consolidation de données enrichies).

### Impacts

Les impacts en termes d’interfaces sont les suivants :

**Impact #1 -** Cette approche favorise le couplage faible des systèmes, les tiers externes pouvant faire évoluer leur structuration interne (attributs spécifiques) en limitant l’impact sur l’architecture du **SYSTEME**.

**Impact #2 -** Les tiers externes doivent avoir la capacité d’être requêtés afin de fournir le détail d’un élément à afficher dans le **SYSTEME**

**Impact #3 -** La recherche détaillée sur les éléments du lot 5 ne pourra se faire que sur la base des données stockées dans le **SYSTEME**. L’évolution du **SYSTEME** dans le futur pourra chercher à étendre la recherche à des éléments résidents dans les lots externes, une capacité de recherche devra être alors déployée, celle-ci ayant la responsabilité de créer et maintenir un index de recherche transverse interrogeable depuis le **SYSTEME**.

### Compléments

Un chantier **Dictionnaire de données** est initié dans le programme FullBIM afin de gouverner l’ensemble des enjeux autour de la donnée :

* Définition des objets métiers, et de leurs interdépendances
* Cycle de vie des objets métiers
* Emplacement de la donnée (Propriétaire de la donnée)

## Les données du **SYSTEME**

Plusieurs familles d’objets doivent être reçues par le **SYSTEME**

* Les **“assets”** (les actifs)
* Les **“tasks”** (les tâches)
* Les **“resources”** (les personnes et les équipes)
* Les **“organisations”**

Remarque : Il est fait l’hypothèse que les fichiers stockés en GED sont accédés au travers d’un objet auquel ils sont rattachés. Ils ne sont pas directement intégrés dans le **SYSTEME**

### Les **“assets”** (les actifs)

#### Émetteur :

* **FULLBIM - LOT #2 Gestion du patrimoine (GP)**
* **FULLBIM - LOT #3 Gestion de la maintenance (GMAO)**

#### Type d’objets pouvant être remontés dans le **SYSTEME** :

* Équipement
* Patrimoine
* Foncier

#### Données remontées dans le SYSTEME

Remarque : Les élements d’accostage sont décrits ci-dessous à titre d’illustration. Ils sont suceptibles d’évoluer en phase de spécifications techniques. La référence documentaire sur le sujet à consulter est le fichier technique de description de l’interface : (Fichier “Swagger”)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nom** | Type | Obligatoire | Description | Categorie | Exemple |
| **entryInServiceDate** | Date | OUI |  | *asset* | 2002-12-30T06:40:00Z |
| **function** | Chaine de caractère | NON |  | *asset* |  |
| **indice** | Booléen | NON |  | *asset* |  |
| **notation** | Numérique | NON | Pour certains objets, une évaluation est fournie | *asset* |  |
| **serialNumber** | Chaine de caractère | NON |  | *asset* |  |
| **station** | Chaine de caractère | NON | GARE : Traduction a revoir | *asset* |  |
| **thumbnailurl** | Chaine de caractère | NON | Une image permettant d’illustrer l’élément | *asset* |  |
| **type** | Chaine de caractère | NON |  | *asset* |  |
| **originUId** | Chaine de caractère | OUI | Identifiant unique de l’élément dans le système d’origine | *core* |  |
| **originparentUId** | Chaine de caractère | OUI |  | *core* |  |
| **originFunctionalId** | Chaine de caractère | NON | Identifiant logique (fonctionnel) de l’objet. Il pourrait remonter plusieurs instances du même objet (par exemple différentes versions) | *core* |  |
| **origin** | Chaine de caractère | OUI | Système d’origine/Émetteur de la donnée | *core* | SCANPRINT |
| **originlastupdated** | Timestamp | OUI |  | *core* | 2002-12-30T06:40:00Z |
| **entity** | Chaine de caractère | OUI | un identifiant de l’entité d’origine | *core* | Cofiroute; ASF; Escota |
| **universalUId** | Chaine de caractère | NON | Identifiant universel de l’objet. Il identifie l’objet dans tous les systèmes (par exemple une référence universellement partagée pour un objet, un fichier de hachage pour un document) | *core* |  |
| **title** | Chaine de caractère | OUI | Le titre de l’élément, aligné sur la langue | *core* |  |
| **description** | Chaine de caractère | OUI | La description courte de l’élément | *core* |  |
| **status** | Chaine de caractère | OUI | Le statut (codification) de l’élément dans le système d’origine | *core* |  |
| **version** | Chaine de caractère | NON | La version de l’élément dans le système d’origine | *core* |  |
| **category** | Chaine de caractère | OUI | Un élément de catégorisation de l’élément (par exemple les “types d’asset” | *core* |  |
| **language** | Chaine de caractère | OUI | La langue de référencement de l’élément | *core* | fr |
| **way** | chaine de caractère | NON |  | *core/location* |  |
| **direction** | Chaine de caractère | NON |  | *core* |  |
| **firstLandmark** | Chaine de caractère | NON |  | *core* |  |
| **geoObject** | Objet JSON | OUI | Un objet permettant de définir un objet géographique associé. Au format geoJSON, il permet de définir un point, une poly-ligne, ou un polygone. | *core* |  |
| **highwayName** | chaine de caractère | NON |  | *core* |  |
| **landmark** | chaine de caractère | NON |  | *core* |  |
| **lastLandmark** | chaine de caractère | NON |  | *core* |  |
| **territoire** | Chaine de caractère | NON |  | *core* |  |
| **town** | chaine de caractère | NON |  | *core* |  |

### Les **“tasks”** (les tâches)

* Visite
* Constat
* Intervention

#### Émetteur :

* **FULLBIM - LOT #2 Gestion du patrimoine (GP)**
* **FULLBIM - LOT #3 Gestion de la maintenance (GMAO)**

#### Données remontées dans le SYSTEME

Remarque : Les élements d’accostage sont décrits ci-dessous à titre d’illustration. Ils sont suceptibles d’évoluer en phase de spécifications techniques. La référence documentaire sur le sujet à consulter est le fichier technique de description de l’interface : (Fichier “Swagger”)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nom** | Type | Obligatoire | Description | Categorie | Exemple |
| **originUId** | Chaine de caractère | OUI | Identifiant unique de l’élément dans le système d’origine | *core* |  |
| **originparentUId** | Chaine de caractère | OUI |  | *core* |  |
| **originFunctionalId** | Chaine de caractère | NON | Identifiant logique (fonctionnel) de l’objet. Il pourrait remonter plusieurs instances du même objet (par exemple différentes versions) | *core* |  |
| **origin** | Chaine de caractère | OUI | Système d’origine/Émetteur de la donnée | *core* | SCANPRINT |
| **originlastupdated** | Timestamp | OUI |  | *core* |  |
| **entity** | Chaine de caractère | OUI | un identifiant de l’entité d’origine | *core* | Cofiroute; ASF; Escota |
| **universalUId** | Chaine de caractère | NON | Identifiant universel de l’objet. Il identifie l’objet dans tous les systèmes (par exemple une référence universellement partagée pour un objet, un fichier de hachage pour un document) | *core* |  |
| **title** | Chaine de caractère | OUI | Le titre de l’élément, aligné sur la langue | *core* |  |
| **description** | Chaine de caractère | OUI | La description courte de l’élément | *core* |  |
| **status** | Chaine de caractère | OUI | Le statut (codification) de l’élément dans le système d’origine | *core* |  |
| **version** | Chaine de caractère | NON | La version de l’élément dans le système d’origine | *core* |  |
| **category** | Chaine de caractère | OUI | Un élément de catégorisation de l’élément (par exemple les “types d’asset” | *core* |  |
| **language** | Chaine de caractère | OUI | La langue de référencement de l’élément | *core* | fr |
| **assignedTeam** | Chaine de caractère | NON | L’équipe en charge de réaliser la tâche | *task* | Tem001 |
| **assignedUser** | Chaine de caractère | NON | L’utilisateur en charge de la tâche | *task* | john\_doe@mycompany.fr |
| **duedate** | timestamp | NON | Date d’échance | *task* | 2002-12-30T06:40:00Z |
| **priority** | Chaine de caractère | NON | Priorité de la tache à effectuer | *task* | 1, 2, 3 |

## Services externes requis dans les tiers d’origine pour requêtage par le **SYSTEME**

Dans une stratégie de formaliser un écosystème où le détail des données stockées dans le SYSTÈME est rapatrié à la volée afin d’assurer une fraicheur optimale à la donnée, différents services doivent être exposés par les systèmes externes (les noms des services sont fournis à titre d’exemple pour simplifier la compréhension)

* *getAscendants (****me****):* Cette fonction permet de récupérer l’ascendance (le père avec/sans ses ancêtres) de l’objet en question.
* *getDescendants(****me****):* Cette fonction permet de récupérer la descendance (la filiation directe et/ou totale) de l’objet en question.
* *getSiblings(****me****):* Cette fonction permet de récupérer la fratrie de l’objet
* *getDetail(****me****):* permet de récupérer le descriptif de l’objet (dont le point de débranchement)
* *getDocuments(****me****):* permet de récupérer la liste des documents (avec url) de l’objet
* *getPK(****highwayID,StartPK, EndPK****)*: permet de récupérer la description du linéaire entre 2 points kilométriques. Ce service sert notamment à l’établissement de la vision synoptique.

Remarque: ***me*** : Correspond à l’identification de l’objet stocké dans le **SYSTEME**. Cela peut être :

* un identifiant **unique propre au système externe** (la clé primaire dans ce système). Il identifie 1 élément dans 1 système externe. (originUId)
* un identifiant **logique** (fonctionnel) au système externe. Il identifie plusieurs versions de l’élément dans le système externe (par exemple pour suivre son évolution). (originFunctionalId)
* un identifiant **universel**. Il identifie l’élément dans tous les systèmes. (universalUId).

## Les services exposés par le **SYSTEME**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID | CONSOMMATEUR (Emetteur de la requête) | PRODUCTEUR (Producteur du service) | Type | Technologie | Nom du service | Description | Nombre de sollicitations par heure | Taille de la requête en ko | Taille de la reponse en ko |
| 2 | FULLBIM - LOT #2 Gestion du Patrimoine (GP) | SYSTÈME | Synchrone | REST API | *objectFeeder#2* | Envoi des éléments nécéssaires et suffisants vers le SYSTÈME. L’hypothèse est que cette alimentation se fait au fil de l’eau (pas de batch) | 10 | 500 | 1 |
| 3 | FULLBIM - LOT #3 Gestion de la maintenance (GMAO) | SYSTÈME | Synchrone | REST API | *objectFeeder#3* | Envoi des éléments nécéssaires et suffisants vers le SYSTÈME. L’hypothèse est que cette alimentation se fait au fil de l’eau (pas de batch) | 10 | 500 | 1 |

Voir le SWAGGER du services d’ingestion (Export pdf) <https://bitbucket.org/sxdva/lot5-hyperviseur/src/master/ArchitectureDossier/tools/07.Interfaces/LOT5-ArchitectureDossier-INTERFACES.pdf>

## Les composants (systèmes externes) à interfacer

* **FULLBIM - LOT #2 Gestion du patrimoine (GP)**
* **FULLBIM - LOT #3 Gestion de la maintenance (GMAO)**
* **FULLBIM - LOT #4 Système d’information géographique (SIG)**
* **FULLBIM - LOT #5 Gestion Electronique de Documents (GED)**
* VINCI Autoroutes - **Identity provider**
* VINCI Autoroutes- **Analyse des données**

### FULLBIM - LOT #2 **Gestion du Patrimoine (GP)**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID | CONSOMMATEUR (Emetteur de la requête) | PRODUCTEUR (Producteur du service) | Type | Technologie | Nom du service | Description | Nombre de sollicitations par heure | Taille de la requête en ko | Taille de la reponse en ko |
| 4 | SYSTÈME | FULLBIM - LOT #2 Gestion du Patrimoine (GP) | Synchrone | REST API | *getDetail* | Pour un utilisateur : Nombre d’objets en **détail** à afficher | 30 | 1 | 10 |
| 5 | SYSTÈME | FULLBIM - LOT #2 Gestion du Patrimoine (GP) | Synchrone | REST API | *getAscendants* | Pour un utilisateur : Nombre **d’ascendance** d’objets à afficher | 30 | 1 | 5 |
| 6 | SYSTÈME | FULLBIM - LOT #2 Gestion du Patrimoine (GP) | Synchrone | REST API | *getSiblings* | Pour un utilisateur : Nombre de **fratries d’objets** à afficher | 7,5 | 1 | 5 |
| 7 | SYSTÈME | FULLBIM - LOT #2 Gestion du Patrimoine (GP) | Synchrone | REST API | *getDescendants* | Pour un utilisateur : Nombre de **descendance** d’objets à afficher | 7,5 | 1 | 5 |
| 24 | SYSTÈME | FULLBIM - LOT #2 Gestion du Patrimoine (GP) | Synchrone | SSO | *getPK* | Affichage du synoptique | 30 | 1 | 10 |

### FULLBIM - LOT #3 **Gestion de la maintenance (GMAO)**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID | CONSOMMATEUR (Emetteur de la requête) | PRODUCTEUR (Producteur du service) | Type | Technologie | Nom du service | Description | Nombre de sollicitations par heure | Taille de la requête en ko | Taille de la reponse en ko |
| 13 | SYSTÈME | FULLBIM - LOT #3 Gestion de la maintenance (GMAO) | Synchrone | REST API | *getDetail* | Pour un utilisateur : Nombre d’objets en **détail** à afficher | 30 | 1 | 10 |
| 14 | SYSTÈME | FULLBIM - LOT #3 Gestion de la maintenance (GMAO) | Synchrone | REST API | *getAscendants* | Pour un utilisateur : Nombre **d’ascendance** d’objets à afficher | 30 | 1 | 5 |
| 15 | SYSTÈME | FULLBIM - LOT #3 Gestion de la maintenance (GMAO) | Synchrone | REST API | *getSiblings* | Pour un utilisateur : Nombre de **fratrie d’objets** à afficher | 7,5 | 1 | 5 |
| 16 | SYSTÈME | FULLBIM - LOT #3 Gestion de la maintenance (GMAO) | Synchrone | REST API | *getDescendants* | Pour un utilisateur : Nombre de **descendance** d’objets à afficher | 7,5 | 1 | 5 |

### FULLBIM - LOT #4 **Système d’information géographique (SIG)**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID | CONSOMMATEUR (Emetteur de la requête) | PRODUCTEUR (Producteur du service) | Type | Technologie | Nom du service | Description | Nombre de sollicitations par heure | Taille de la requête en ko | Taille de la reponse en ko |
| 12 | SYSTÈME | FULLBIM - LOT #4 Système d’information géographique (SIG) | Synchrone | REST API | *MAPS* | Pour un utilisateur : Nombre d’objets en **détail** à afficher | 30 | 1 | 100 |

### FULLBIM - LOT #5 **Gestion Electronique des Documents**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID | CONSOMMATEUR (Emetteur de la requête) | PRODUCTEUR (Producteur du service) | Type | Technologie | Nom du service | Description | Nombre de sollicitations par heure | Taille de la requête en ko | Taille de la reponse en ko |
| 21 | SYSTÈME | FULLBIM - LOT #5 Gestion Electronique de Documents | Synchrone | REST API | *getDocuments* | Pour un utilisateur : Nombre de documents associés à un objet à afficher (URL retourné par le LOT 6, ouverture du document depuis le poste de l’utilisateur) | 3 | 5 | 1 |

### VINCI Autoroutes - **Identity provider**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID | CONSOMMATEUR (Emetteur de la requête) | PRODUCTEUR (Producteur du service) | Type | Technologie | Nom du service | Description | Nombre de sollicitations par heure | Taille de la requête en ko | Taille de la reponse en ko |
| 22 | SYSTÈME | VINCI Autoroutes - Identity provider | Synchrone | SSO | *authenticate* | Nombre de demande d’authentification des utilisateurs VINCI Autoroutes. Hypothèse: correspond aux utilisateurs courant de la plateforme | 206 | 1 | 5 |

### VINCI Autoroutes- **Analyse des données**

A définir

## Cartographie des Flux

### Hypothèses

Sur la base des exigences non fonctionnelles en termes de capacité, le diagramme de flux suivant peut etre défini:

# Architecture générale

## Définition

Cette vue d’ensemble a pour objectif de synthétiser les grands thèmes d’architecture de **Data Factory**.

## Approche architecturale

Architecture : vue d’ensemble

Architecture : vue d’ensemble

Afin d’assurer la pérennité une approche architecturale déclinant la plateforme en

* Un sous-système capturant la donnée, agissant comme un référentiel. On parle de système référentiel ou **“System Of Record”**
* Un système permettant d’interagir avec les utilisateurs. On parle alors de système d’engagement des utilisateurs ou **“System Of Engagement”**
* Un système permettant de manipuler les données collectées afin d’en tirer une valeur ajoutée. On parle alors de **“System of Insight”**

Il sera fait la promotion d’un couplage faible entre les différents systèmes par l’implémentation d’une approche **micro services** reposant sur des API

La force de cette stratégie est de permettre

* une atomicité permettant une évolutivité plus simple (pas d’effets silo)
* des cycles de vie des solutions adaptés à la typologie du système à implémenter (Agilité, vélocité des SoE, Pérennité, Stabilité des SoR )

Dans le cadre du Lot 5,

* La partie de « Collecte des données » (dans un format “pivot” métier de référence) aura un rôle de SoR,, alimenté par une couche d’intégration avec le monde extérieur permettant d’assurer la médiation avec les systèmes externes au moyen de format pivot générique.
* La partie au contact de l’utilisateur aura un rôle de SoR
* il n’y a pas de « System of Insight ». Néanmoins la BI externe à alimenter répond a cette description.

# Modèle de composants

## Vue statique

### Paquets et Composants

L’hyperviseur peut être décomposé en sous-systèmes et représenté dans un diagramme de paquets présentant les dépendances entre paquets.

#### Paquets

|  |  |
| --- | --- |
| **Paquet** | Description |
| **Client WEB** | Ensemble des composants mis en place sur le client WEB (navigateur internet) de l’utilisateur. Par exemple : Rafraichissement asynchrone, Récupération de données, Parallélisation de traitements. |
| **Front-End (Serveur) WEB** | Ensemble des composants mis en place sur le serveur pour connecter les clients WEB : Contenu graphique, données… |
| **Services ‘Metier’** | Ensemble des composants implementant les services métier: Récupération/Insertion de la donnée, processus métier |
| **Stockage des données** | Composant permettant la sauvegarde des données |
| **Integration Tiers externes** | Mécanisme d’exposition de la plateforme pour s’interfacer avec des tiers externes |

#### Composants

Composants logiciels

Composants logiciels

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Paquet** | Composant | Détail | Rôle | Responsabilités |
| *Services ‘Métier’* | **Sécurité (Univers, Profils et Permissions)** | [Sécurité](./0501.ComponentsModel.md) | Gestion des profils et permission de la plateforme | Récupère le profil de l’utilisateur et les permissions associées. |
| *Services ‘Métier’* | **Audit et Traçabilité** | [Tracabilité](./0502.ComponentsModel.md) | Gestion des évènements “métier” de la plateforme | Expose un mécanisme de capture des évènements “métier” de la plateforme à des fins de traçabilité |
| *Accès aux données* | **Journalisation** | [Journalisation](./0503.ComponentsModel.md) | Gestion des évènements techniques de la plateforme | Mécanisme de journalisation des évènements |
| *Client WEB* | **Web Client Manager** |  | Gestion de la page coté “Client” | Maitrise de la construction et de la mise a jour de la page affichée, Traitement des évènements envoyés par la GUI, Récupération des données auprès du composant DataManager |
| *Client WEB* | **Web Client GUI Renderer** |  | Affichage de la page | Affichage de la page Web, Envoi des évènements (Appels asynchrones de rafraichissement…) au WebClient Mgr |
| *Client WEB* | **Web Client DataManager** |  | Accès aux données sur le client | Envoi et récupération des données à afficher (structure, ressources statiques, données “métier”) |
| *Front-End (Serveur) WEB* | **Web Resource Manager** |  | Gestion des ressources à envoyer au client | Fourni au client Web les resources statiques et dynamiques pour construire la page, ainsi que les données “métier” (bidirectionnelle) |
| *Front-End (Serveur) WEB* | **Static Web Resource Manager** |  | Gestion des ressources statiques | Gestion des ressources statiques (image, icone, feuille de style) |
| *Front-End (Serveur) WEB* | **Dynamic Web Resource Manager** |  | Gestion des ressources dynamiques | Gestion des ressources dynamiques (structure de la page) |
| *Services ‘Métier’* | **Business Service Geospatial Data Viz" as BS1** |  | Univers Hyperviseur | Logique “métier” de l’univers Hyperviseur |
| *Services ‘Métier’* | **Component “Business Service DashBoard and Report” as BS2** |  | Univers Dashboard | Logique “métier” de l’univers Dashboard |
| *Accès aux données* | **Interface - Business Events** |  | Mise en œuvre d’un format Pivot |  |
| *Accès aux données* | **Interface - Actifs** |  | Mise en œuvre d’un format Pivot |  |
| *Accès aux données* | **Interface - Taches** |  | Mise en œuvre d’un format Pivot |  |
| *Accès aux données* | **Interface - Orga** |  | Mise en œuvre d’un format Pivot |  |
| *Stockage des données* | **Data - Business Events** |  | Persistence des données | Stockage des évènements utilisateurs de la plateforme |
| *Stockage des données* | **Data - Business Data** |  | Persistence des données | Stockage des données “métier” (récupérées des autres lots, générées par le SYSTÈME) |
| *Stockage des données* | **Data - Configuration** |  | Persistence des données | Stockage des données de configuration |
| *Stockage des données* | **Data - Technical (log…)** |  | Persistence des données | Stockage des données techniques (journalisation) |
| *Integration Tiers externes* | **LOT #1 - Demi Interface - Ingestion (feeder)** |  | Exposition API | API d’ingestion des données externes issues du Lot#1 |
| *Integration Tiers externes* | **LOT #2 - Demi Interface - Ingestion (feeder)** |  | Exposition API | API d’ingestion des données externes issues du Lot#2 |
| *Integration Tiers externes* | **LOT #3 - Demi Interface - Ingestion (feeder)** |  | Exposition API | API d’ingestion des données externes issues du Lot#3 |
| *Integration Tiers externes* | **LOT #4 - Demi Interface - Ingestion (feeder)** |  | Exposition API | API d’ingestion des données externes issues du Lot#4 |
| *Integration Tiers externes* | **LOT #GED - Demi Interface - Ingestion (feeder)** |  | Exposition API | API d’ingestion des données externes issues du Lot GED |

Remarque : Certain composants mentionnés ne sont pas représentés dans le schema pour en simplifier la lecture

## Vue dynamique

### Diagramme de processus

### Diagramme de séquence

# Système de transformation

Les transformations font partie de la famille **System of Insight**. Une transformation permet de produire une donnée de type Y à partir d’une donnée de type X.

### Anatomie d’une transformation

Une transformation est un **microservice** autonome pouvant être déclenché par :

* Un appel HTTP

Ce microservice représente un noeud de transformation de l’ensemble des transformations disponible sur la plateforme. Un noeud de transformation peut executer un seul type de transformation.

Les trasnformations sont déclenché en mode **Fire and Forget**:

* L’appel permet de poster une tâche dans la file d’attente interne au **noeud** de transformation.
* Une tache doit être identifié dans le plateforme avant son arrivée dans le noeud de transformation.
* Les tâches sont ensuite traités dans leurs ordre d’arrivée dans le noeud de transformation (**FIFO - First In First Out**) en tâche de fond.
* Un noeud peut executer plusieurs tâches en concurrence.

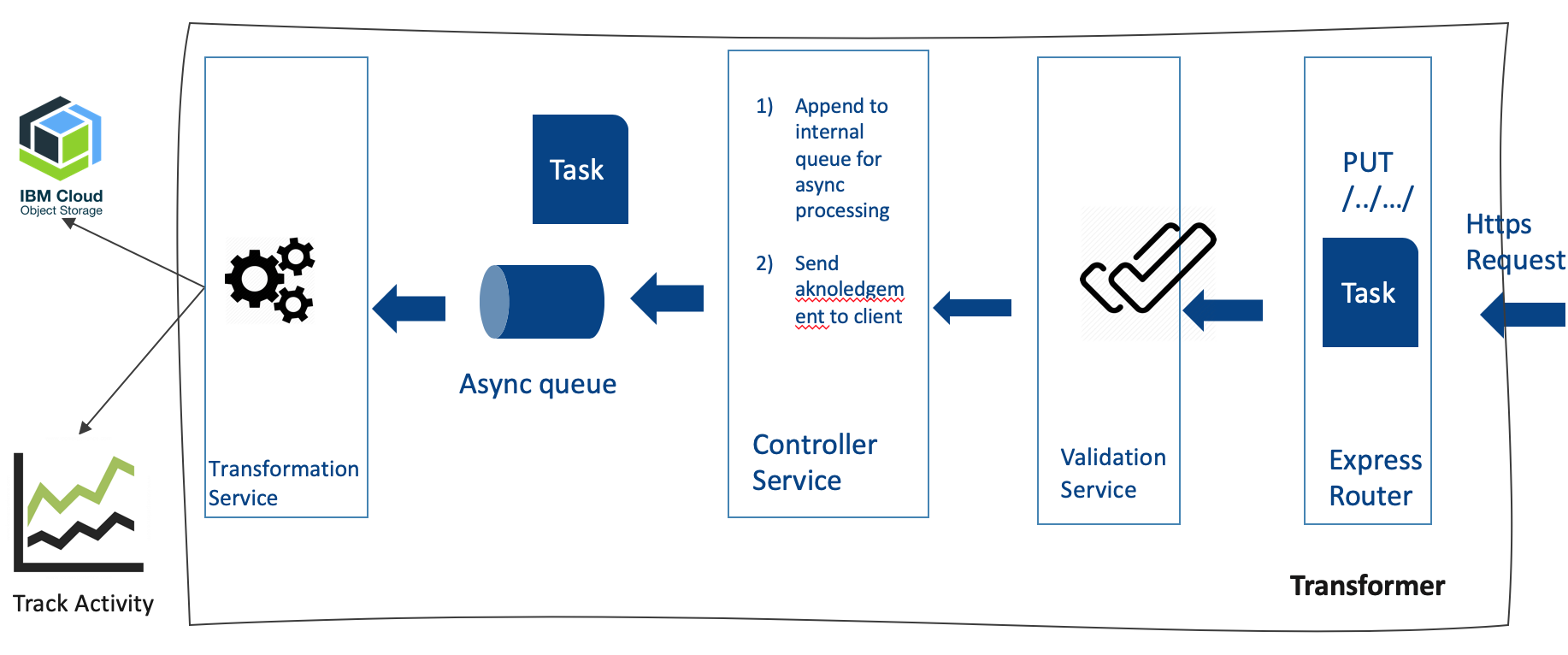
Les consommateurs (**Apps Beyond**) appellent une transformation avec un ensemble d’informations décrivant:

* La localisation du fichier source de transformation
* La destination ou seront placés le/les fichier(s) résultant(s)
* Accéssoirement des paramétres spécifique à la transformation

Plus techniquement, l’objet source d’une transformation doit au moins contenir:

* **id**: l’identifiant du job de transformation
* **srcBucket**: Le bucket contenant le fichier source
* **srcFile**: Le chemin du fichier source
* **destBucket**: Le bucket destination où le fichier résultat de transformation sera enregistré
* **opts**: Optionnel, un ensemble d’options/arguments pour enrichir le process d’une transformation
* **weight**: Optionnel, pour de future besoins, le degré de priorité d’une transformation pour la gestion de file d’attente.

Les fichiers produits par un noeud de transformation sont enregistrés sur un bucket **COS**.



Architecture : vue d’ensemble

### Track activity

L’avancement d’un job de transformation doit être loggé dans le **track activity** de la plateforme **BEYOND**. Le track activity est en charge de fournir un ensemble de services permettant à des tenants de récupérer le statut d’une transformation par son identifiant.

### Distribution des noeuds de transformation

Les noeuds de transformations sont distribués par types de transformations :

* Un noeud = une transformation
* Chaque noeud est taggé son type de transformation.
* Chaque type de transformation posséde un répartisseur de charge (**point d’entrée vers une transformation**) permettant d’acheminer les requêtes vers les noeuds du même type. Ainsi nous intégrons aussi une gestion santé des noeuds, lorsqu’un noeud ne répond plus, il est ignoré.

Cette architecture nous permet d’isoler chaque type de transformation tout en gardant une homogénéité dans l’exploitation de celles-ci.

### Integration dans BEYOND

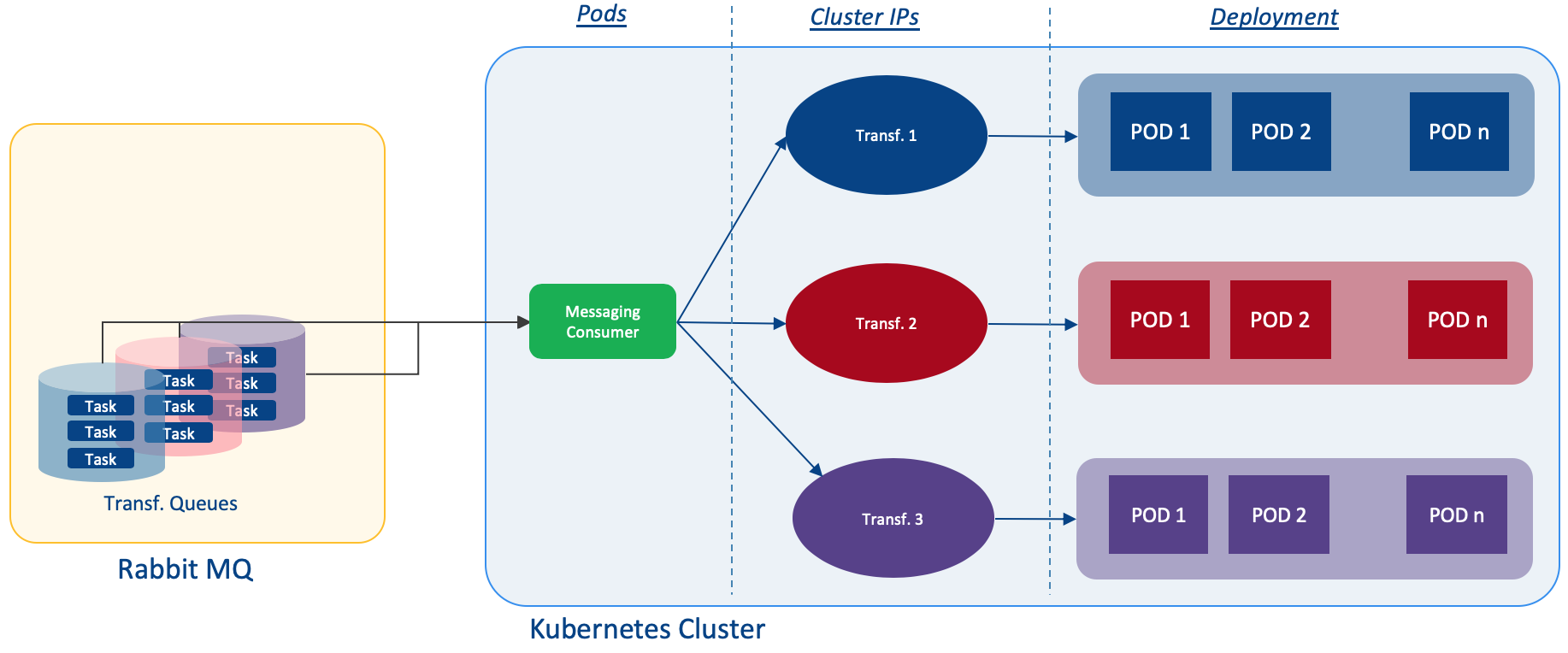
Pour garder un couplage faible entre les tâches de transformations et le reste de la plateforme, un outils de messagerie (**RabbitMQ**) sera utilisé pour permettre aux tenants de poster des tâches de transformation.

Les tâches doivent être identifiées d’une manière unique et libellé (type de transf.) avant de les poster sur l’outil de messagerie. ceci nous permet de :

* Récupérer l’avancement d’une tâche avec le **Track activity** (en mode Pull).
* Router un job vers la bonne queue

De ce fait, La mise en place des queues de transformation se fera en suivant les principes suivant:

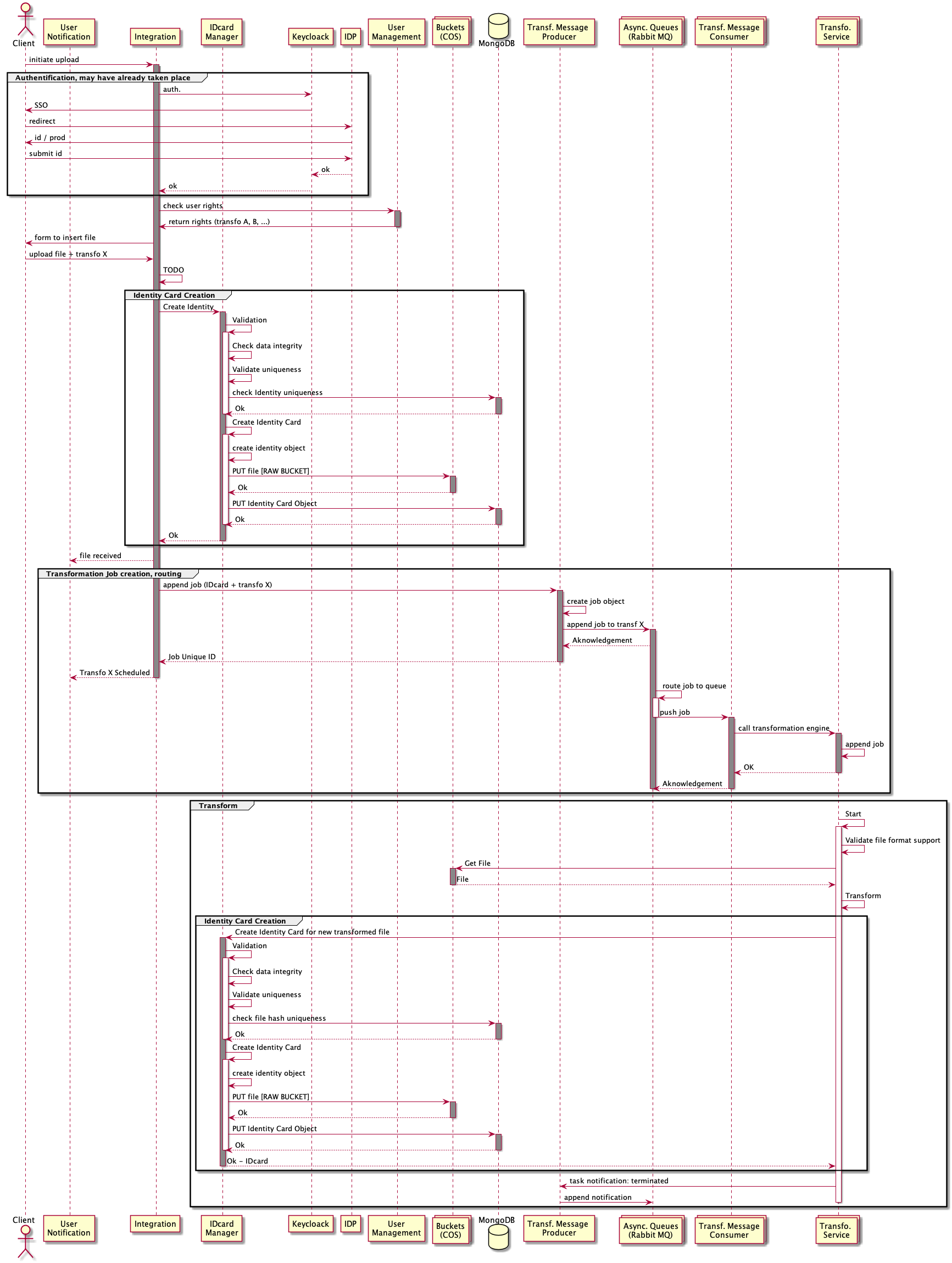
* Le **message producer** déverse dans un *exchanger* de la manière suivante:
  + l’exchanger doit être de type **direct**
  + le producer est responsable de créer les “messages” (représentant les tâches) et générer un identifiant unique pour chaque message.
  + le producer envoie des messages avec une **clé de routage** correspondant au type de transformation.
* Le **message consumer** récupére depuis les *queues* de transformation de la maniére suivante:
  + le consumer s’assure de la configuration des queues de transformation et notamment leurs liaison avec l’exchanger (le routage à base d’une clé spécifique vers une queue de transformation).
  + le consumer dispose d’un agent connecté à chaque queue. chaque agent est responsable de déliverer les messages d’une transformation vers le point d’entrée de la transformation.
  + chaque queue est durable, de cette maniére si le consumer se déconnecte les tâches non traité persistent dans la queue.
    - Si le consumer n’arrive pas à joindre une transformation, il doit mettre le job dans une **dead letter queue**.



Architecture : vue d’ensemble

### Injection d’un job de transformation

Le diagramme de séquence suivant décrit les interractions entre les différents composants de la plateforme **BEYOND** pour lancer un job de transformation.



## AD-BYD-DF-001 : Unicité des fichiers stockés dans la datafactory lors d’un import

### Décision d’architecture

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Identifiant | Décision d’architecture | Domaine | Date | Comité |
| **AD-001** | Alternative 02 - Pas d’unicité des fichiers entrants | Data | Juillet 2019 | Décision PO Plateforme |

### Enjeux

Le **SYSTEME** doit permettre l’acquisition de fichiers depuis des systèmes tiers autorisés, soit au travers d’un canal “Machine à Machine”, soit au travers d’une interface de remontée manuelle de fichiers.

L’enjeux à maitriser est la responsabilité du **SYSTEME** comme un “point de vérité” ou de référence en termes de données pour ses consommateurs (tenants).

### Hypothèses

Par conception, le **SYSTEME** permet d’acquérir des fichiers. La suppression de ceux-ci est logique (positionnement d’un état). Il n’y a pas de suppression physique en mode nominal (Mécanisme d’hibernation des données froides à définir), afin de garantir la traçabilité au cas où le fichier aurait été utilisé. L’accès aux fichiers est protégé par une liste de contrôle d’accès.

### Alternatives

#### Alternative 01 - Garantie de l’unicité des fichiers entrants

Chaque (version d’un) fichier ne peut être remontée qu’une seule fois dans le **SYSTEME**. La vérification se fait par comparaison de l’empreinte (hash) du fichier à remonter avec ceux stocké dans les ‘ID Cards’

**Avantages :**

* Garantie de l’unicité, du positionnement du SYSTEME comme un référentiel.
* Promotion du partage de l’information entre les tenants. 2 tiers utilisant le même fichier sont assurés de reposer sur la même ‘instance’ du fichier et donc de partager les informations complémentaires (Les annotations par exemple).
* Intégrité : La même information n’est pas dupliquée. La traçabilité est plus forte (Tous les utilisateurs de l’information pointe sur les même données)
* Ergonomie : Le moteur de recherche ne remontera pas plusieurs instances d’un même fichier.
* Sécurité : Dans le cadre d’une remontée M2M, cela éviterait un effet de bord consistant en l’envoi massif d’un même fichier qui ne serait pas rejeté (malveillant: Déni de service, ou pas: Anomalie dans un tiers externe qui repousserait plusieurs fois le même fichier par erreur ).

**Inconvénients :**

* Gestion des droits de visibilité sur les fichiers : Si un tiers veut accéder un fichier (potentiellement inaccessible du fait de ses droits), l’administrateur métier du SYSTEME devra lui fournir les accès

#### Alternative 02 - Pas d’unicité des fichiers entrants

Chaque (version d’un) fichier peut être remonté plusieurs fois dans le **SYSTEME**

Remarque : Cette alternative intègre la gestion de l’unicité d’un fichier pour un tenant.

**Avantages :**

* Souplesse d’utilisation

**Inconvénients :**

* cf. avantages : Alternative 01.
* Volumétrie

### Recommandation

Alternative 01 - Garantie de l’unicité des fichiers entrants

### Justification

Cette alternative présente les qualités suivantes :

* L’intégrité des données de la plateforme et de leur utilisation. ‘Point de vérité’.
* Sécurité -Performance

### Implications

N/A

### Décisions d’architecture liées

N/A

# Modèle de déploiement

## Liste des URLs

## Technologies

### Clusters

### Composants

### Dépots de code

# Modèle de données

## Data Collections

### objectIDCard

Carte d’identité des objets du **SYSTEME**

* [Schéma](./schemas/0801.ObjectIDCard.json)
* [Exemple](./schemas/0801.ObjectIDCardexample.json)

### Enrichissement des IDCards

[Template](./schemas/0802.Enrichment_Template.schema.json)

* Un template sert à contrôler et identifier les attributs d’un enrichissement. Il est à l’origine de la saisie d’un enrichissement donné.
  + Si un template est introuvable, l’enrichissement ne peut pas être modifié. Cependant il peut être visualisé/supprimé
* Un template peut être modifié après sa création. Une autre révision de celui-ci est créée.
* Un template peut être verrouillé par son créateur/admin. Les autres utilisateurs et/ou système ne pourront pas le modifier/supprimer.
* Un template dispose d’un statut permettant d’activer/désactiver un template
* Pour un attribut d’un template, il est possible de décrire la permission nécessaire pour maintenir cette information (éditer). Ainsi il est possible de répartir les taches de saisie d’un enrichissement sur plusieurs profiles ou de restreindre l’ensemble de modification à un seul profil.

[Enrichissement](./schemas/0802.Enrichment.schema.json)

* Un enrichissement est un ensemble d’informations saisie par un utilisateur/systeme à base d’un template.
* Un enrichissement est lié à un idCard

### Annotation d’un objet

Une annotation d’un objet du **SYSTEME** permet d’indexer les objets selon une expression (une étiquette). Il existe 2 types d’annotation :

* les annotations “publiques” sont communes à l’ensemble de la plateforme ; elles sont utilisées dans les recherches transverses mais aussi pour typer certains objets (voir ci-dessous les associations). On distinguera ainsi les annotations fonctionnelles des annotations techniques. La liste des annotations publiques est gérée au niveau de l’administration de la plateforme, afin de garantir le maintient d’un ontologie couvrant un périmètre global standard.
* les annotations “privées” sont propres au Tenant propriétaire de la données ; elles permettent une recherche sur un périmètre / prisme spécifique, non couvert par l’ontologie des annotations publiques.

[Référence des annotations Publiques](./schemas/0803.Annotation.Reference.schema.json)

### Association Entre objets

Une association entre objets du **SYSTEME** permet de gérer les liens entre objets. Ces liens sont “typés” pour permettre de proposer différentes associations. Ces associations peuvent être fixées explicitement (un utilisateur associe des objets) ou bien implicitement (une opération sur un objet implicant la création d’autres objets conduit à la création de la relation associée ; par exemple dans le cadre d’une transformation).

Une association est une IDCard possédant l’annotation “association”. Les objets membres de l’association sont identifiés dans un enrichissement de l’association.

On distinguera deux types d’association :

* les relations hiérarchiques (l’objet A est l’antécédant de l’objet B qui est l’antécédant de l’objet C)
* les relations de composition (l’objet A contient les objets B et C).

[Exemple association](./schemas/0804.Association.example.json)

[Exemple association composée](./schemas/0804.Association.enrichment.group-example.json)

[Exemple association hierarchique](./schemas/0804.Association.enrichment.hierarchy-example.json)

# Liste des logiciels

## Gouvernance des composants et de leurs versions

Les projets doivent mettre a jour les composants en suivant les directives suivantes :

* Un composant ne peut pas être ajouté au référentiel sans l’accord de la Direction Technique SIXENSE. Un dossier de sélection doit être formalisé au moyen du modèle suivant :

[Ajout d’un composant à la plateforme BEYOND](https://bitbucket.org/sxdibm/byd-all-documentation/src/master/00.Qualite/01.PlanQualiteTechnique/modeles-document/ajout-cots/01.cots-selection.md)

* La version majeure d’un composant ne peut être mise à jour sans l’accord de la Direction Technique SIXENSE.
* Les versions mineures et patch doivent être mises à jour en fonction des livraisons de la solution.

## Fichier de référence

La liste des composants est maintenu dans le fichier suivant avant d’être exportée dans ce document :

[Fichier MS-Excel](https://bitbucket.org/sxdibm/byd-all-documentation/src/master/01.byd-DataFactory/1.ArchitectureDossier/artifacts/02.cots-list/BYD-DF_COTSLists_v01.00.xlsx)

## Composants

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nom** | Version | URL | Licence | Description |
| **app-root-path** | 2.*.* | https://www.npmjs.com/package/app-root-path | MIT | This simple module helps you access your application’s root path from anywhere in the application without resorting to relative paths like require(“../../path”). |
| **bluebird** | 3.*.* | https://www.npmjs.com/package/bluebird | MIT | Bluebird is a fully featured promise library with focus on innovative features and performance |
| **body-parser** | 1.*.* | https://www.npmjs.com/package/body-parser | MIT | Node.js body parsing middleware. |
| **cookie-parser** | 1.*.* | https://www.npmjs.com/package/cookie-parser | MIT | Parse Cookie header and populate req.cookies with an object keyed by the cookie names. |
| **cors** | 2.*.* | https://www.npmjs.com/package/cors | MIT | CORS is a node.js package for providing a Connect/Express middleware that can be used to enable CORS with various options. |
| **dotenv** | 6.*.* | https://www.npmjs.com/package/dotenv | BSD-2-Clause | Dotenv is a zero-dependency module that loads environment variables from a .env file into process.env |
| **express** | 4.*.* | https://www.npmjs.com/package/express | MIT | Express.js, or simply Express, is a web application framework for Node.js, released as free and open-source software under the MIT License. |
| **helmet** | 3.*.* | https://www.npmjs.com/package/helmet | MIT | Helmet helps you secure your Express apps by setting various HTTP headers. |
| **jwt-decode** | 2.*.* | https://www.npmjs.com/package/jwt-decode | MIT | jwt-decode is a small browser library that helps decoding JWTs token which are Base64Url encoded. |
| **lodash** | 4.*.* | https://www.npmjs.com/package/lodash | MIT | The Lodash library exported as Node.js modules. |
| **log4js** | 3.*.* | https://www.npmjs.com/package/log4js | MIT | This is a conversion of the log4js framework to work with node |
| **mime** | 2.*.* | https://www.npmjs.com/package/mime | MIT | A comprehensive, compact MIME type module. |
| **Node** | 10.*.* | https://nodejs.org/en/ | MIT | Node.js is a JavaScript runtime built on Chrome’s V8 JavaScript engine. |
| **passport** | 0.*.* | https://www.npmjs.com/package/passport | MIT | Passport’s sole purpose is to authenticate requests, which it does through an extensible set of plugins known as strategies. |
| **passport-jwt** | 4.*.* | https://www.npmjs.com/package/passport-jwt | MIT | A Passport strategy for authenticating with a JSON Web Token. |
| **pg** | 7.*.* | https://www.npmjs.com/package/pg | MIT | Non-blocking PostgreSQL client for Node.js. Pure JavaScript and optional native libpq bindings. |
| **PostGIS** | 2.*.* | https://postgis.net/ | GPLv2 | PostGIS is a spatial database extender for PostgreSQL object-relational database. It adds support for geographic objects allowing location queries to be run in SQL. |
| **PostgreSQL** | 9.\* | https://www.postgresql.org/ | PostgreSQL License | PostgreSQL, is an object-relational database management system (ORDBMS) with an emphasis on extensibility and standards compliance |
| **sequelize** | 4.*.* | https://www.npmjs.com/package/sequelize | MIT | Sequelize is a promise-based Node.js ORM for Postgres, MySQL, SQLite and Microsoft SQL Server. It features solid transaction support, relations, read replication and more. |
| **swagger-ui-express** | 3.*.* | https://www.npmjs.com/package/swagger-ui-express | MIT | API documentation generator |
| **useragent** | 2.*.* | https://www.npmjs.com/package/useragent | MIT | Useragent originated as port of browserscope.org’s user agent parser project also known as ua-parser. |

|**yamljs**|0.*.*|https://www.npmjs.com/package/yamljs|MIT|Standalone JavaScript YAML 1.2 Parser & Encoder. Works under node.js and all major browsers. Also brings command line YAML/JSON conversion tools.|