# Extension du logiciel Actes Office par intégration dans une architecture SOA

Charlotte Huchet

le cnam

**CONSERVATOIRE NATIONAL DES ARTS ET METIERS**

**CENTRE REGIONAL ASSOCIE DE MIDI-PYRENEES**

---

**MÉMOIRE**

**présenté en vue d'obtenir**

**le DIPLOME d'INGENIEUR CNAM**

**SPECIALITE : INFORMATIQUE**

**OPTION : Architecture et Ingénierie des Systèmes et des Logiciels (AISL)**

**Par**

**HUCHET Charlotte**

---

# EXTENSION DU LOGICIEL ACTES OFFICE PAR INTEGRATION DANS UNE ARCHITECTURE SOA

**Soutenu le 22 octobre 2015**

---

**JURY**

**PRESIDENT : M. Y. POLLET, Professeur des Universités, CNAM Paris**

**MEMBRES : M. H. BATATIA, Maîtres de conférences, INP Toulouse**

**M. T. MILLAN, Maître de conférences, UPS Toulouse**

**M. L. IVETON, Tuteur – Ingénieur Développement, Berger-Levrault**

**M. J. ORIANO, Responsable production, Berger-Levrault**

# REMERCIEMENTS

J'adresse mes remerciements aux personnes qui m'ont aidé dans la réalisation de ce mémoire.

En premier lieu, je remercie M. Batatia. En tant que Directeur de mémoire, il m'a guidé dans mon travail et m'a aidé à trouver des solutions pour avancer.

Je remercie également l'équipe pédagogique du CNAM, ainsi que tous mes camarades de formation.

Je remercie aussi M. Iveton, mon tuteur au sein de l'entreprise Berger-Levrault, qui m'a aidé en me soutenant dans la rédaction de ce mémoire.

Je remercie Mme. Perrigaud, avec qui j'ai travaillé au sein de Berger-Levrault, qui m'a aidé en me fournissant les informations nécessaires, et en ayant toujours les mots justes pour m'aider.

Je remercie également toutes les personnes qui ont travaillé avec moi sur mon projet, qui ont toujours su répondre à mes questions : Mathieu Duval, Philippe Sirgue, Mathieu Passenaud, José Capelle et tous les autres que j'aurais pu oublier.

Un merci particulier à mon compagnon qui m'a soutenu dans tous les moments clés de ma formation.

# ABREVIATIONS

ESB : Entreprise Service Bus

TDT : Tiers De Télétransmission

@CTES : Aide au Contrôle de légaliTé dématErialiSé

BLES : Berger-Levrault Echange Securise

# TABLE DES MATIERES

# I INTRODUCTION

L'architecture SOA est de plus en plus utilisée dans les entreprises. Cette Architecture Orientée Service, dont le terme est apparu au cours de la période 2000-2001, est depuis un modèle au sein des systèmes d'information et dans l'informatique en général.

C'est une forme d'architecture de médiation, modèle d'interaction applicative qui met en œuvre des services. Ceux-ci ont une forte cohérence interne, par l'utilisation d'un format d'échanges pivot, mais également des couplages externes lâches, en utilisant une couche d'interface interopérable.

La mise en place de ce genre d'architecture permet de facilité l'extension logicielle, sujet récurrent dans le monde du génie logiciel. En effet, toute application a pour vocation d'être étendue pour des besoins fonctionnels.

Si cette question n'a pas été posée en amont, la gestion de cette extensibilité devient donc un problème majeur. C'est le sujet qui m'a été posé sur le logiciel actes-office lors de mon année de travail. Ce logiciel souffrait d'une perte de connaissance et avait des besoins d'interopérabilité.

Une des solutions à envisager est d'intégrer ce logiciel dans une architecture lui permettant d'isoler ses besoins de services. Le service devient donc externe et factorisable pour d'autres besoins. Pour un éditeur de logiciel cette factorisation est un gain de temps énorme, puisque le service n'est plus dépendant de l'applicatif.

Au cours de mon année de travail au sein de la société Berger-Levrault il m'a été demandé d'intégrer le logiciel Actes-office à ce type d'architecture. En effet, les besoins d'Actes office au niveau notamment de la télétransmission des actes administratifs, est fortement dépendante d'un tiers. Son extension dépend donc de son interopérabilité avec ces derniers. De plus, la mise en place de cette architecture bénéficiera à l'ensemble des logiciels développés par l'entreprise.

La problématique posée est celle de l'extension logicielle. Plus particulièrement, comment étendre un logiciel par la gestion de l'interopérabilité ? Nous tacherons d'y répondre en démontrant que l'architecture SOA répond à cette attente.

Nous présenterons, pour commencer, le problème d'extension logicielle en lui-même, en présentant le contexte dans lequel il s'est posé. Puis nous définirons l'architecture tout en montrant comment la prise en compte de l'interopérabilité par l'intégration dans une architecture SOA résout le problème d'extension. Ensuite la mise en application de ces principes sera détaillée afin de mettre en lumière les apports pour le logiciel Actes-office. Pour finir sur un bilan de cette architecture, les faiblesses et les intérêts qu'ils pourront avoir sur l'entreprise.

# II PROBLEME D'EXTENSION LOGICIELLE

## II.1 ACTIVITES DE L'ENTREPRISE BERGER-LEVRAULT

Berger-Levrault est la plus ancienne maison d'édition française, dont les origines remontent à 1463. Depuis toujours Berger-Levrault travaille aux côtés du monde public et l'accompagne dans ses évolutions. Spécialisée dans l'édition papier, cette longue histoire a plus récemment évolué par l'arrivée dans le groupe de plusieurs entités issues de l'édition électronique : Magnus, Sedit Marianne, Convergence, Ecolesoft, DIS, Progor, Sofiac, Segilog, Aducits et dernièrement Intuitive, Coba et Sigems. En rassemblant les atouts et les talents de ces entreprises, Berger-Levrault, filiale depuis près de 10 ans du Groupe Accueil, propose aujourd'hui une offre unique à l'usage du monde public et de la santé :

- Logiciels et matériels informatiques
- Ouvrages métiers, documents et formulaires réglementaires
- Services d'accompagnement

Actuellement Berger-Levrault est une société dont le siège se trouve à Paris. Grâce à sa culture de création, Berger-Levrault est le 1$^{er}$ éditeur multicanal d'informations et de solutions de gestion en Europe. Alliant conseils, produits, services et formations, son offre donne les moyens nécessaires aux intervenants du secteur public, de la santé et de la société civile de se consacrer à leur cœur de métier.

Avec près de 60 000 clients et plus de 1 000 collaborateurs, le groupe Berger-Levrault est acteur de l'administration numérique pour des usages publics plus efficaces et respectueux de l'humain.

Rapprocher les femmes, les hommes et leurs administrations, est la vocation de Berger-Levrault, partie prenante de tous les temps forts de la vie citoyenne (état civil, santé, enseignement, social, entreprise, justice, élections...) à l'heure de l'ouverture massive des données publiques en Europe.

Logiciels, services, ouvrages et formulaires, Berger-Levrault propose une offre globale aux acteurs des collectivités, de l'administration et de la santé. Aujourd'hui, mille collaborateurs accompagnent au quotidien près de 40000 collectivités

publiques, établissements de santé et médicaux sociaux en leur garantissant des solutions rigoureuses, performantes et innovantes. Berger-Levrault réalise un chiffre d'affaires de 108 M€ en 2013.

Maintenant, Berger-Levrault compte plusieurs établissements dans toute la France : Paris, Lille, Toulouse, La chapelle sur Erdre, Montpellier, Champigneulles, Lyon, Nantes et Colmar. Le personnel est réparti de la façon suivante : 300 informaticiens en R&D et production, 250 formateurs en services d'accompagnement de proximité, 150 spécialistes du commerce et du marketing et 300 techniciens de maintenance pour l'assistance à distance et sur site. L'organigramme ci-dessous présente l'organisme d'activités chez Berger-Levrault.

Les processus au sein de l'entreprise sont scindés en trois : le pilotage, le soutien et la production.

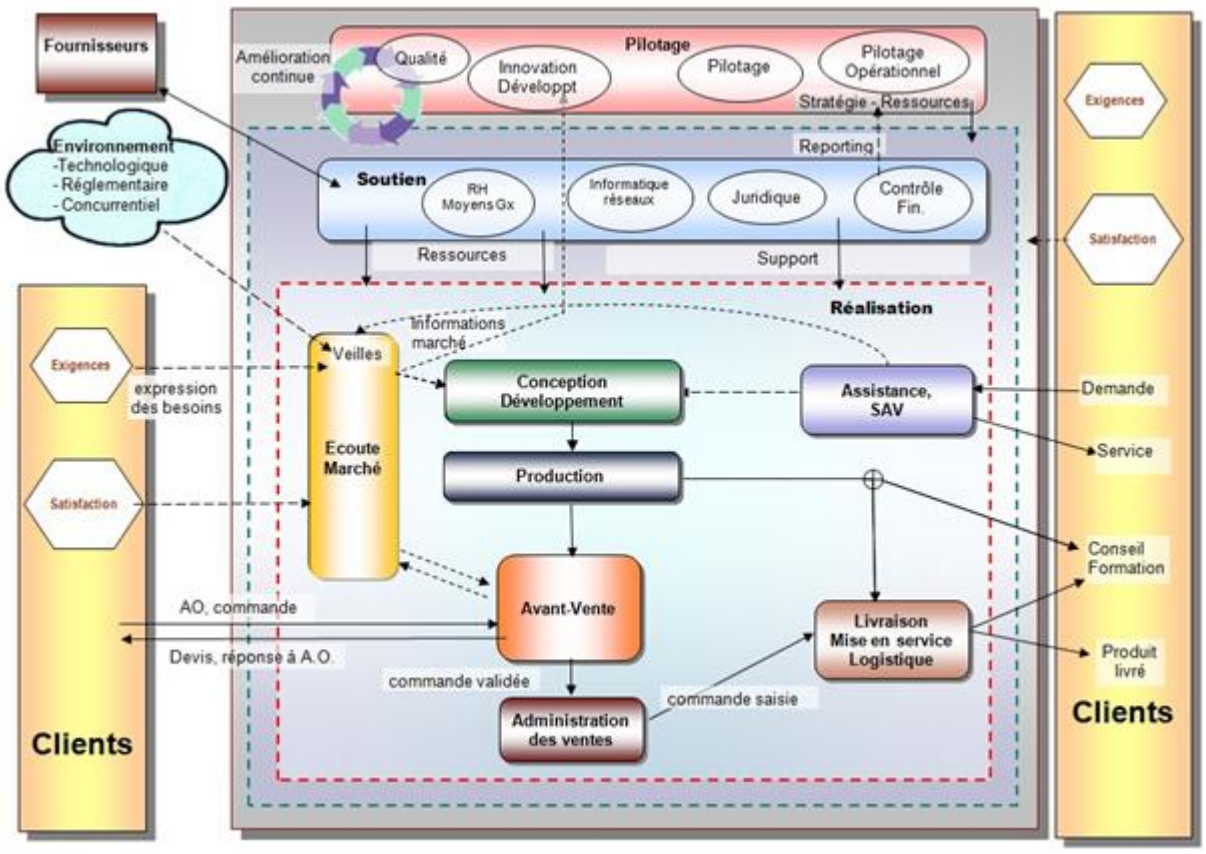


Figure 1 - Cartographie des processus

J'ai pour ma part effectué mon travail dans la partie réalisation, dans l'équipe de conception et de développement, plus particulièrement au sein de l'activité Famille.

## II.2 ACTES OFFICE

Actes Office est un logiciel permettant aux mairies ou aux conseils généraux de préparer et de suivre les conseils municipaux ou des assemblées. Il permet également la rédaction des arrêtés et des délibérations. Ce logiciel propose la gestion des phases de décisions, de convocation des élus et les votes des délibérations.

Actes Office est interfacé avec les principales solutions de transmission dématérialisées des actes en préfecture, pour effectuer simplement et en toute sécurité l'envoi des documents via le module AoLink. Il est également interfacé à un parapheur permettant la signature électronique des actes.

### II.2.1 FONCTIONNALITES

Les principales contraintes du logiciel sont :

- Ne pas alourdir le traitement quotidien du service responsable de la gestion et de la préparation des actes administratifs avec des fonctions trop complexes ou inutiles.
- S'adapter aux besoins et aux habitudes de travail de chaque organisation et de chaque service.

Les fonctions d'Actes Office sont les suivantes :

- Paramétrage pour adapter Actes Office aux habitudes des organisations.
- Des fonctions « courantes » pour permettre de préparer, suivre et gérer les événements (réunions et conseils) et les documents relatifs à ces événements (Rapport, Ordre du jour, Convocation, Note de synthèse, Délibération, Compte-rendu…).


Figure 2-Page d'accueil d'Actes Office

Deux étapes principales sont prises en compte dans le produit :

- avant la séance :
    - planification des séances : permet de préparer les séances en enregistrant les informations la concernant. Un retro planning sur les différentes tâches à réaliser pour organiser la séance du conseil permet de mettre en place des alertes.


Figure 3- Retro planning des taches à effectuer

    - planification des commissions : permet de choisir l'ordre du jour et d'y associer les différents projets à évaluer en séance.


Figure 4 - Mise en place de l'ordre du jour

- préparation des documents de pré-séance : génération de l'ordre du jour et publipostage des convocations.


Figure 5 - Convocations à la séance

- après la séance :

- saisie des présences : permet d'enregistrer les présents pour la séance ainsi que les procurations
- saisie des votes : permet d'enregistrer les votes pour chaque sujet


Figure 6 - interface de saisie des votes

- édition des documents définitifs (Compte-rendu, Recueil des actes administratifs…) : l'ensemble des informations enregistrées précédemment permettent de générer les documents officiels nécessaires pour la légalité.
- consultation

### II.2.2 MODULE AOLINK DE DEMATERIALISATION DES ACTES

L'objectif de cet outil est de permettre l'envoi des actes administratifs vers le serveur du Ministère de l'Intérieur pour transmission vers les préfectures et de fournir aux utilisateurs d'**ACTES OFFICE** un moyen simple de gérer la télétransmission des actes soumis au contrôle de légalité.


Figure 7 - interface utilisateur AoLink

## II.2.2.1 FONCTIONNALITES

- Préparation des actes avant envoi : Un acte est transmis au module aolink depuis actes office, il est cependant nécessaire d'effectuer des vérifications au préalable afin de vérifier la conformité avec le protocole acte.

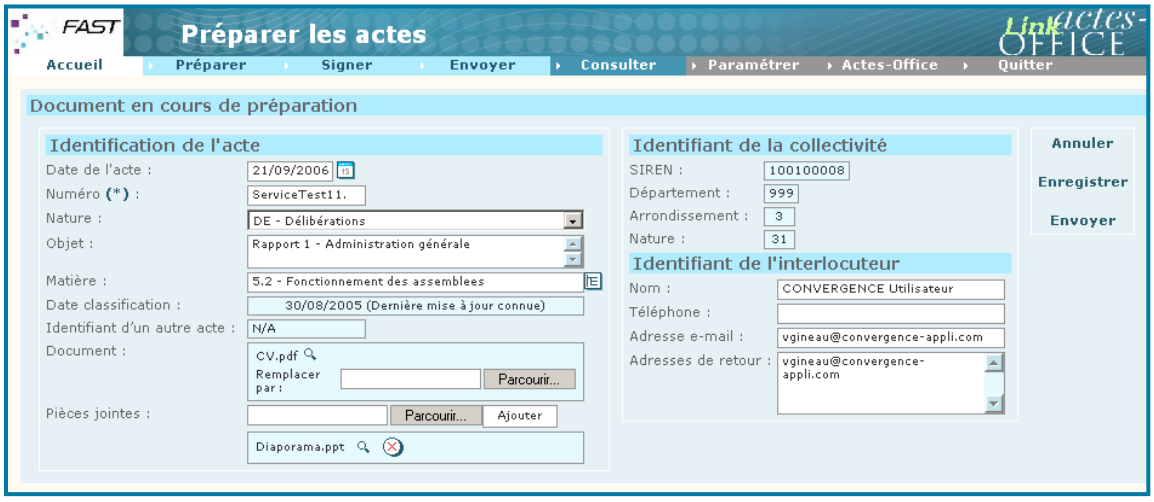


Figure 8 - Préparation d'un acte

- Signature d'un acte : les actes peuvent être signés numériquement avant l'envoi au tdt par l'utilisation d'un parapheur.

- Envoi : L'envoi à proprement parlé du document au tiers de télétransmission. Cette action peut être faite pour chaque acte ou par lot.

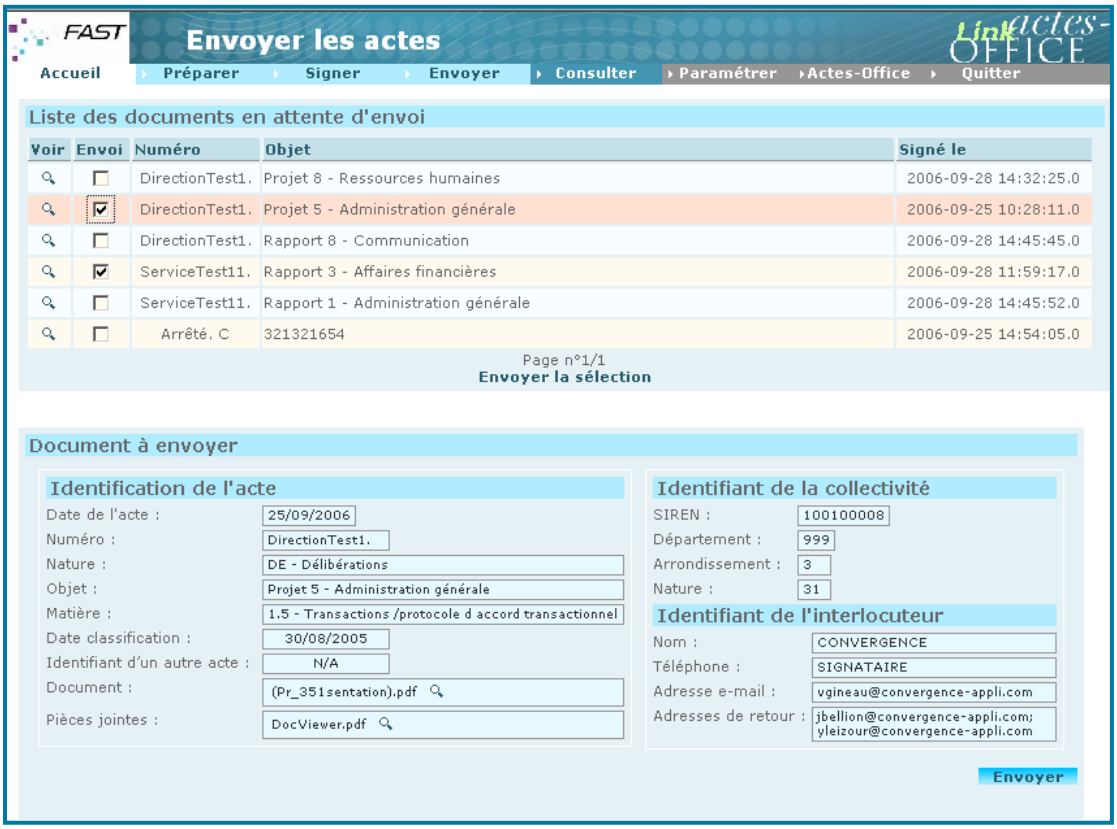


Figure 9 - Interface d'envoi des actes

- Consultation de l'état de la télétransmission : permet de visualiser l'état d'avancement de l'acte dans son processus d'envoi au serveur du ministère de l'intérieur.


Figure 10 - interface de consultation des actes

.

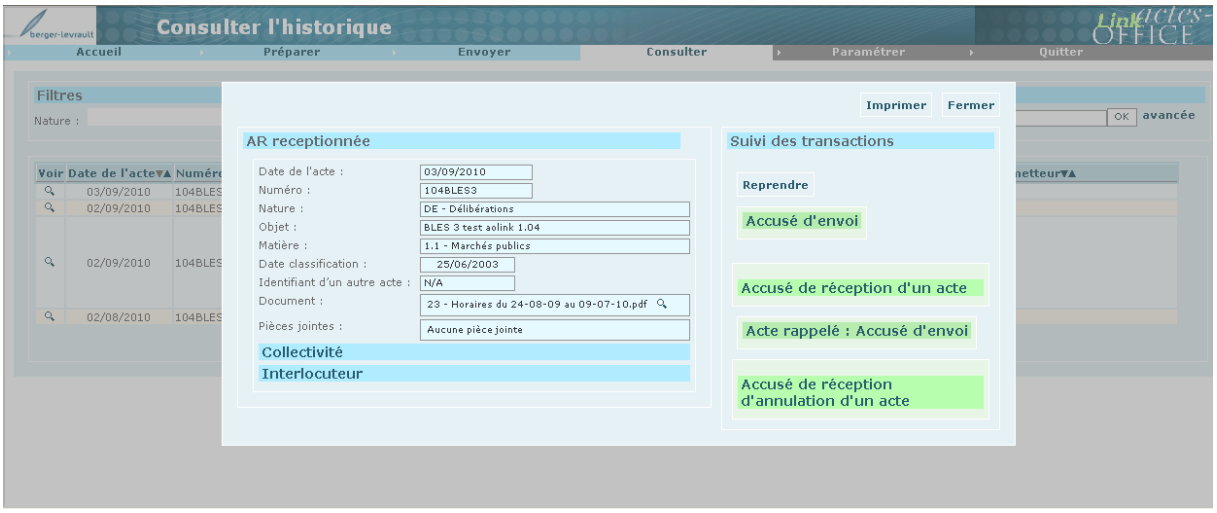


**Figure 11 - visualisation des documents relatifs à la validation par le tdt**

Chaque passage dans une phase du processus génère un document de validation ou de refus, ils doivent être accessibles afin de suivre clairement l'avancement de l'envoi.

- Le rappel d'un acte : l'envoi d'un acte doit à tout moment pouvoir être annulé et être rappelé pour que la validation s'arrête.

- Paramétrage : gestion des utilisateurs, gestion des certificats permettant d'accéder aux tdt, mise à jour de la classification (la classification est spécifique à chaque préfecture et permet de classifier le type des actes envoyés cf annexe 1)

### II.2.3 LIMITATIONS

Le logiciel actes office est un logiciel dont le premier développement date de 2004, de nombreuses équipes se sont succédées sur son développement et son évolution. L'entreprise Berger-Levault a subit sur celui-ci, notamment sur le module AoLink, une perte de connaissance. Les évolutions de ce modules ont donc au cours du temps été mises en exergue.

La logique spécifique liée aux besoins des tdt a donc été perdue. Le besoin d'interfacer Acte office avec un nouveau tdt dans l'optique d'un appel d'offre a donc fait ressurgir le besoin d'interopérabilité de celui-ci avec les tdt.

Il a également été mis en évidence que chaque développement spécifique afin d'interfacer actes office avec un nouveau tdt engendre un surcout lié au développement des caractéristiques spécifique de celui-ci.

Une enquête auprès des clients a permis de définir des fonctionnalités attendues dans ce genre de logiciel de gestion. La fonctionnalité liée à la Ged a donc été priorisée car Actes office génère énormément de document, les utilisateurs bénéficierai ainsi d'une gestion des documents optimisée.

### II.2.4 BESOINS D'INTEROPERABILITE

A la fois les fonctionnalités liées à la ged mais également la gestion des multiples tdt peuvent être regroupés en un besoin principal : l'interopérabilité de l'application avec ses nombreux services.

#### II.2.4.1 INTEROPERABILITE AOLINK-TDT

Le module AoLink ayant subi une perte de compétence au sein de l'entreprise il est donc important d'effectuer un retro engineering sur cette application afin de définir la faisabilité de la mise en application d'une nouvelle architecture permettant l'interopérabilité de la connexion au tdt et d'actes office. L'envoi des documents aux tdts est défini par le protocole @actes

##### II.2.4.1.1 LE PROTOCOLE @CTES

@CTES, qui signifie « Aide au Contrôle de légaliTé dématErialiSé », désigne d'une part le programme visant à développer un système d'information ayant pour objectif la dématérialisation de la transmission des actes soumis au contrôle de légalité et budgétaire.

@ctes désigne également l'application permettant aux agents de préfectures et des sous-préfectures de contrôler les actes soumis à l'obligation de transmission aux services en charge du contrôle de légalité télétransmis par les collectivités territoriales, leurs établissements publics locaux (EPL) et les établissements de coopération intercommunale (EPCI), via un système d'information fourni par un opérateur de transmission.

Ce projet s'inscrit dans le cadre de la modernisation de l'Etat à laquelle il contribue par le développement de l'e-administration.

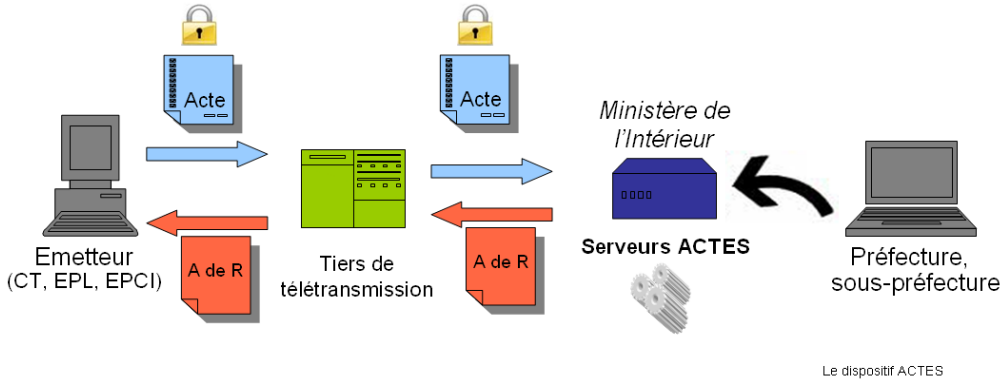


Figure 12 - @CTES sources collectivites-locales.gouv.fr

Pour les collectivités territoriales, leurs établissements publics locaux et les établissements de coopération intercommunale (EPCI), c'est la possibilité de :

- transmettre instantanément par voie électronique à la préfecture les actes soumis au contrôle de légalité (arrêtés et délibérations avec leurs annexes, contrats, etc.) à tout moment de la journée, avec la possibilité d'annuler un envoi en cas d'erreur ;
- recevoir en temps réel, l'accusé de réception qui rend l'acte exécutoire, sous réserve des formalités de publication et de notification.

Les collectivités territoriales, leurs EPL ou les EPCI qui décident de transmettre par voie électronique tout ou partie de ses actes soumis au contrôle de légalité doivent :

- prendre une délibération autorisant l'exécutif à signer avec le préfet de département une convention relative à la transmission électronique des actes.
- choisir un opérateur de transmission parmi la liste des dispositifs homologués, disponible sur ce portail.
- signer une convention avec le préfet de région.

**Les besoins de dématérialisation :**

- Accélération des échanges avec la préfecture, et la réception quasi immédiate de l'accusé de réception aux actes transmis
- Entrée en vigueur quasi automatique de l'acte grâce à l'envoi de réception automatique
- Réduction des coûts liés à la transmission électronique des actes à la préfecture et à la réduction corrélative du nombre d'exemplaires imprimés
- Fiabilisation des échanges
- Traçabilité des échanges
- Intégration du contrôle de légalité dans une chaîne de dématérialisation complète et ininterrompue
- Démarche protectrice de l'environnement : la dématérialisation permet de faire face à la croissance du nombre d'actes et à l'augmentation de leur volume

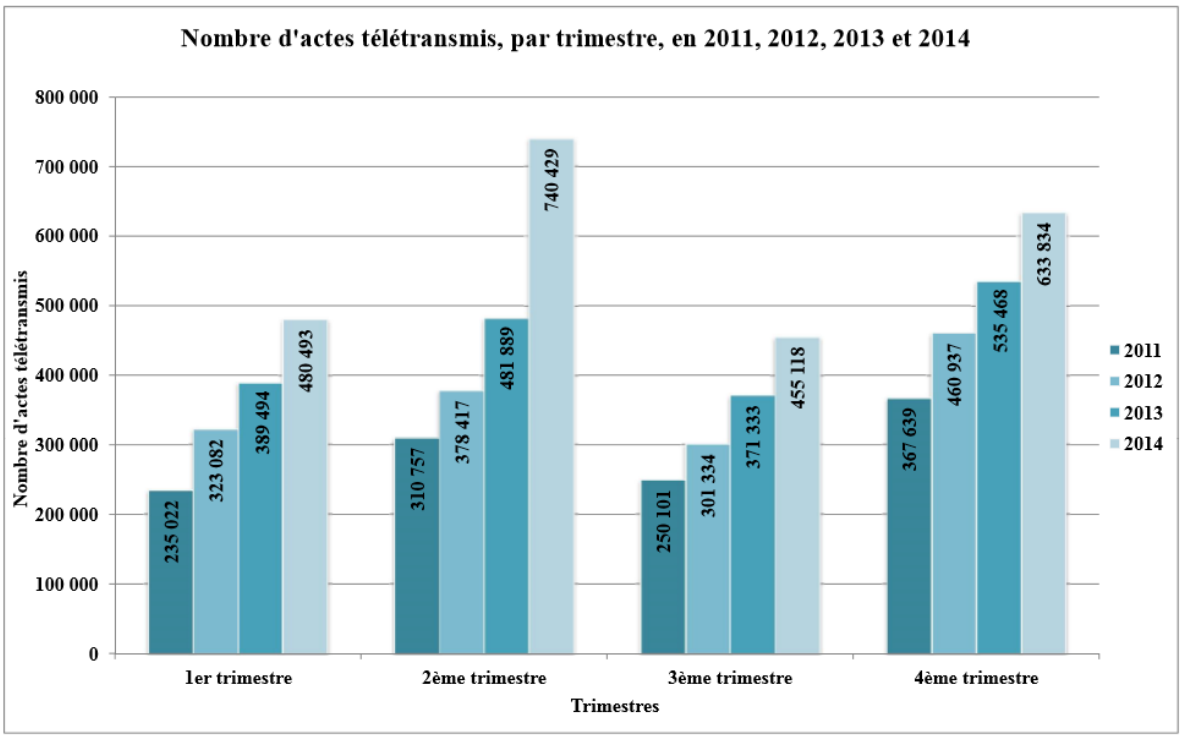


Figure 13 - Nombre d'actes télétransmis - source :collectivites-locales.gouv.fr

Un Tdt pour pouvoir être utilisé doit répondre à un cahier des charges fixé par le ministère de l'intérieur, afin de certifier la qualité des données envoyées. Aux vues

de la forte augmentation du nombre d'actes télétransmis, le nombre de tdt validés a fortement augmenté pour arriver  aujourd'hui à 26.

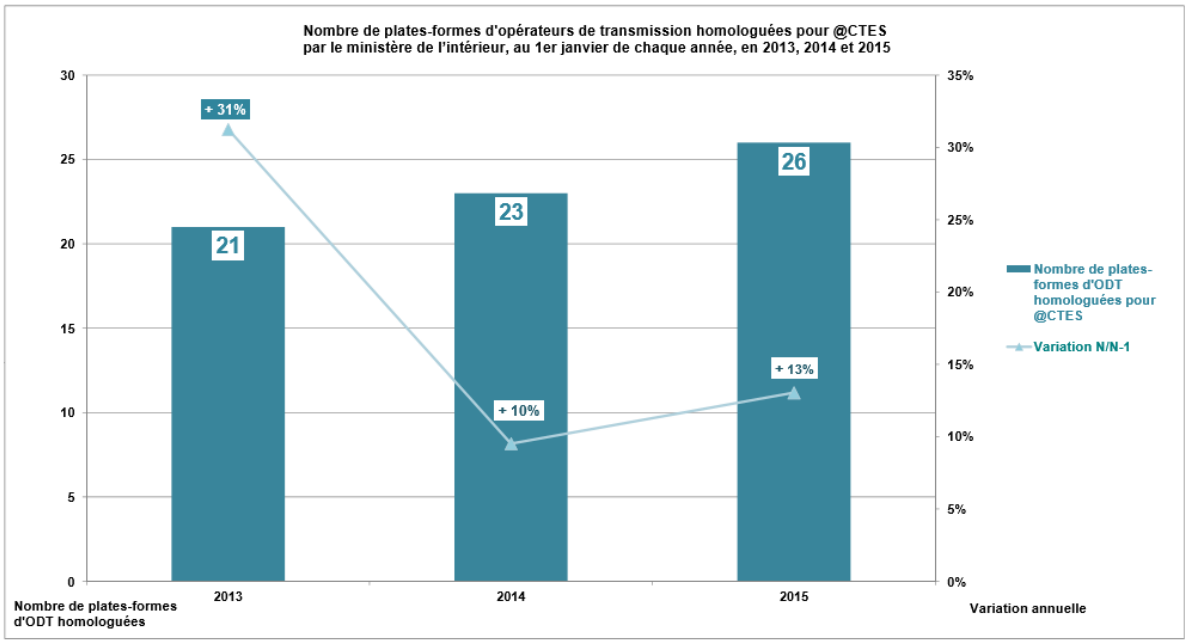


Figure 14 - Variation du nombre de tdt sources: collectivites-locales.gouv.fr

Si Actes office devait s'interfacer à chacun des 26 tdt existant aujourd'hui sur le marché le cout en développement serait trop important, il a donc été décidé de mettre en place une architecture permettant d'interfacer Actes office avec n'importe quel tdt en factorisant le cout de développement. D'autant plus qu'Actes office n'est pas le seul logiciel de la gamme Berger-Levrault permettant la connexion avec ce protocole.

### II.2.4.2 INTEROPERABILITE ACTES OFFICE-GED

Le besoin exprimé par les clients de l'intégration d'une GED avec Actes-office pose également un problème de compatibilité avec la solution de GED déjà utilisée par la collectivité. En effet les entreprises utilisant actes office et éprouvant ce besoin dans actes office possèdent déjà leur propre service de GED. Il est donc important de prendre en considération les solutions qu'ils utilisent. Les quatre GED utilisaient par le client sont Alfresco, SharePoint, Nuxeo et Open Bee. Cependant cette liste n'est pas finie. Contrairement à la fonctionnalité de contrôle dématérialisé des actes

la logique d'interopérabilité avec la Ged peut être réalisée en amont de la conception. C'est pourquoi il a été choisi de mettre en place une architecture de type SOA pour répondre aux deux besoins.

# III INTEROPERABILITE PAR INTEGRATION DANS UNE ARCHITECTURE SOA

## III.1 ARCHITECTURE ACTES-OFFICE

### III.1.1 ARCHITECTURE ACTES-OFFICE ACTUELLE

L'application Actes-office est interfacée avec W4 et le iParapeur dont nous ne nous occuperons pas mais également avec AoLink. Le module AoLink est actuellement interfacé avec trois tiers de télétransmissions : FAST, BLES (Berger-Levrault échanges sécurisés) et S²LOW (Service Sécurisé Libre interopérable pour la Vérification et la Validation).

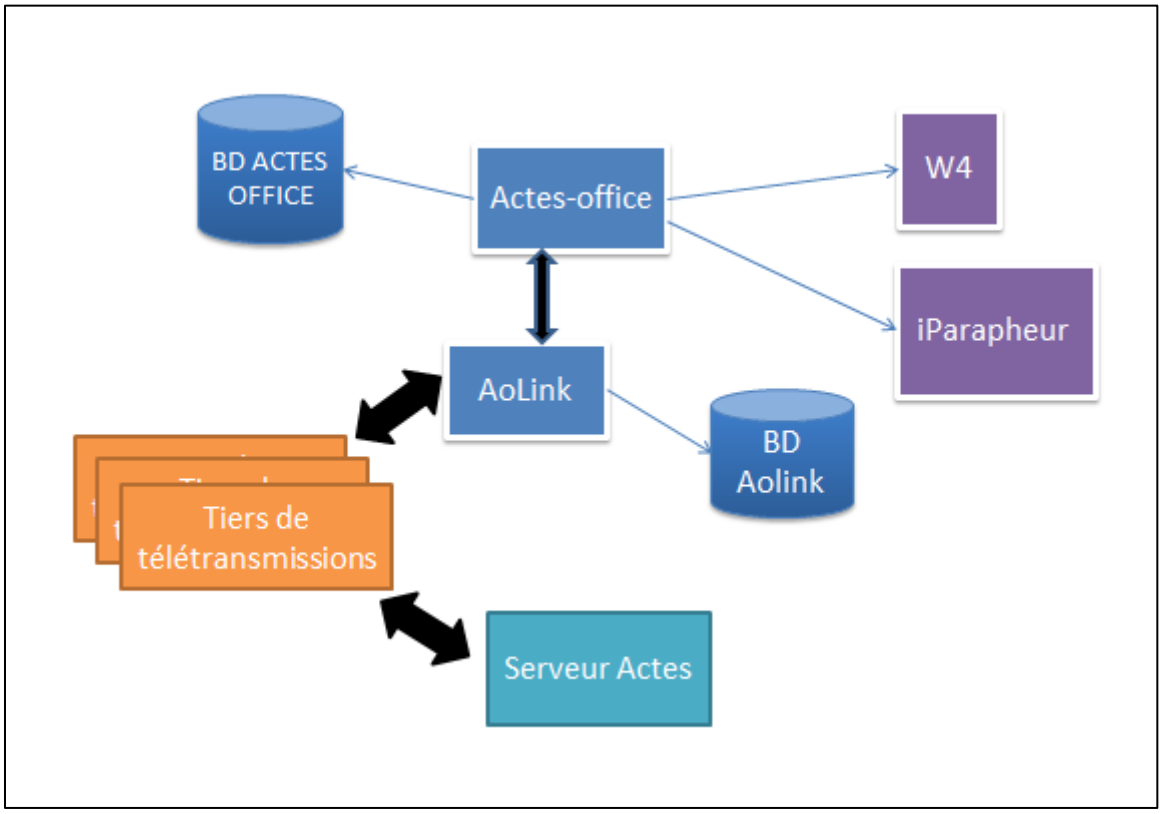


Figure 15 - Architecture Actes-office actuelle

Comme nous pouvons le voir sur le schéma, Actes office est associé à w4 et le iParapheur dont nous ne nous occuperons car ils ne sont pas dépendant des fonctionnalités à développer. Actes-office possède sa base donnée, AoLink aussi ce qui peut paraitre original pour un module technique de l'application. Il faut savoir que

#### III.1.1.1 AOLINK ET LES TIERS DE TELETRANSMISSIONS

AoLink est le module d'Actes office qui permet le contrôle de légalité des actes dématérialisés et est donc interfacé avec trois tiers de télétransmissions. Leurs protocoles sont différents, nous les comparons dans le tableau suivant.

**Tableau 1 - comparatif Tdt**

| Tiers de télétransmissions | protocole | message | certificats |
|---|---|---|---|
| **FAST** | SSL | Enveloppe XML | RGS** |
| **S²LOW** | HTTPS | Paramètre REST | X |
| **BLES** | HTTPS | Paramètre REST | X |

Nous pouvons voir dans le tableau 1 que les trois tiers de télétransmission actuellement utilisés par Actes office utilisent à la fois le même protocole de communication https mais sur certains élément comme le type de messages envoyés ils sont différents. Ce qui rend complexe la mise en place de l'interopérabilité d'AoLink avec les Tdts.

En effet, l'ensemble des 26 tiers de télétransmissions répondent à la fois aux mêmes exigences pour être approuvée par le ministère de l'intérieur mais utilisent en même temps leur propre fonctionnement interne. Ce qui met donc en évidence le besoin d'interopérabilité d'Actes office.

La solution pour pouvoir être compatible avec l'intégralité des Tdts est d'intégrer ce logiciel autour d'une architecture plus globale qui met en place des solutions de services.

### III.1.2 Architecture Actes-Office bus

L'architecture d'Actes office a donc été repensée afin d'être intégré dans une architecture SOA. Les deux besoins de l'applicatif, bien que leurs fonctionnalités soient différentes répondent toutes deux à un problème d'interopérabilité. Une solution a donc été mise en place pour chacun d'eux.

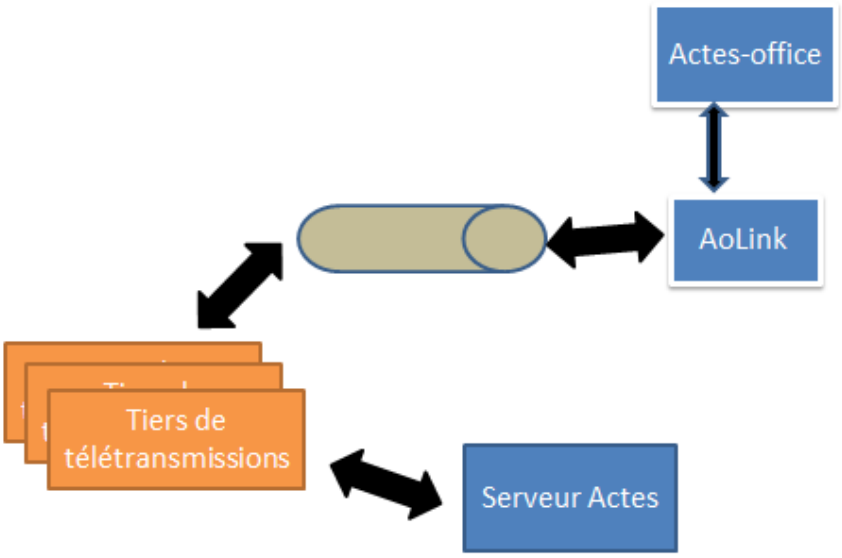


**Figure 16 – Nouvelle architecture Actes-office interopérabilité Tdts**

Comme nous pouvons le voir sur la figure 16 la solution est de mettre en place un composant entre AoLink et les Tiers de télétransmissions afin d'éviter le redéveloppement d'AoLink à chaque ajout de Tdt. C'est donc à ce module que sera donnée la mission de dispatcher vers les Tdts correspondant. L'appel est centralisé

dans AoLink en utilisant un seul protocole, un seul format et une seule façon de s'authentifier.

Pour la question de la GED, il existe un problème similaire, puisque qu'à terme Actes office pourra être utilisé avec n'importe quelle GED sur le marché. Il faut donc mettre en amont une architecture qui permettra a priori l'interopérabilité au niveau de cette fonctionnalité.

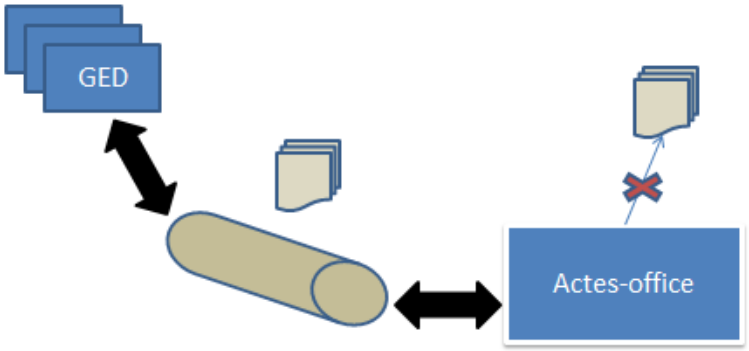


Figure 17 - Nouvelle architecture Actes-office GE

Sur la figure 17, l'architecture de la GED est représentée par un module inter Actes office GED. Ce module permet de dispatcher les actions d'Actes office sur la GED selon celle qui a été paramétrée précédemment.

Nous pouvons observer sur ces deux schémas que les solutions aux deux problèmes posés par actes office ont une résolution similaire. La mise en place d'un module permettant d'exporter le problème d'interopérabilité. C'est à ce module qu'il est déporté. Il est donc envisagé d'intégrer Actes office à une architecture mettant en place ce module lui-même, cette solution est présentée sur la figure 18.

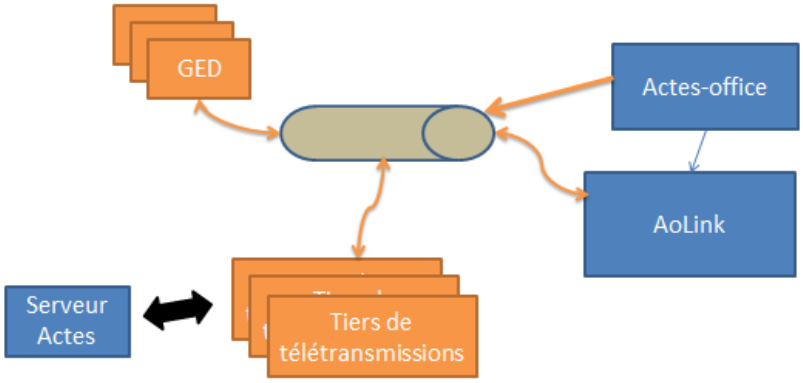


Figure 18 - Architecture Actes office

## III.2 BUS BLES

Le bus BLES est un projet interne à Berger-Levrault permettant de factoriser les développements au niveau des flux de télétransmissions des documents. Il est utilisé pour les flux de documents interne, pour les flux PES mais aussi pour les flux actes. Ainsi lorsqu'un service souhaite s'interfacer avec un nouveau flux, c'est sur le bus BLES qu'il est développé afin de le rendre accessible à n'importe quel autre produit Berger-Levrault.

Son fonctionnement est basé sur des web services avec une communication sécurisée par HTTPS. Ces derniers sont accessibles en REST et en SOAP.

Le diagramme d'état ci-dessous présente les différentes étapes nécessaires à l'envoi vers un tiers de télétransmission.

Chacune de ses étapes doit ainsi être mise en place par l'application Actes office afin d'envoyer un flux actes quel que soit le Tdt souhaité en sorti. Le choix du tiers de télétransmission et la gestion des certificats est alors mise en place au cours du paramétrage de l'utilisateur du bus BLES.

Les étapes de mise en place du bus sont liées à une gestion de dossier. Ce dossier doit tout d'abord être crée afin de créer la base de ce qui va être envoyé. Ce dossier contient les informations du client qui seront nécessaire au Tiers de télétransmission. Ensuite afin de donner les informations au bus BLES il faut effectuer une modification de ce dossier afin de lui envoyer les documents et les informations de classification dont il a besoin. L'étape qui suit est celle de l'envoi en lui-même au Tdt. Cette étape est associée à une série de vérification afin de vérifier si les informations fournies sont compatible avec le Tdt choisi. C'est pourquoi il faut toujours envoyer toutes informations disponibles.

Finalement le retour du document signé et validé et accessible depuis ce dispositif. D'autres étapes intermédiaires sont disponible, comme le rappel d'un acte en cas d'erreur client ou le retour tdt en cas d'erreur au niveau du tdt.

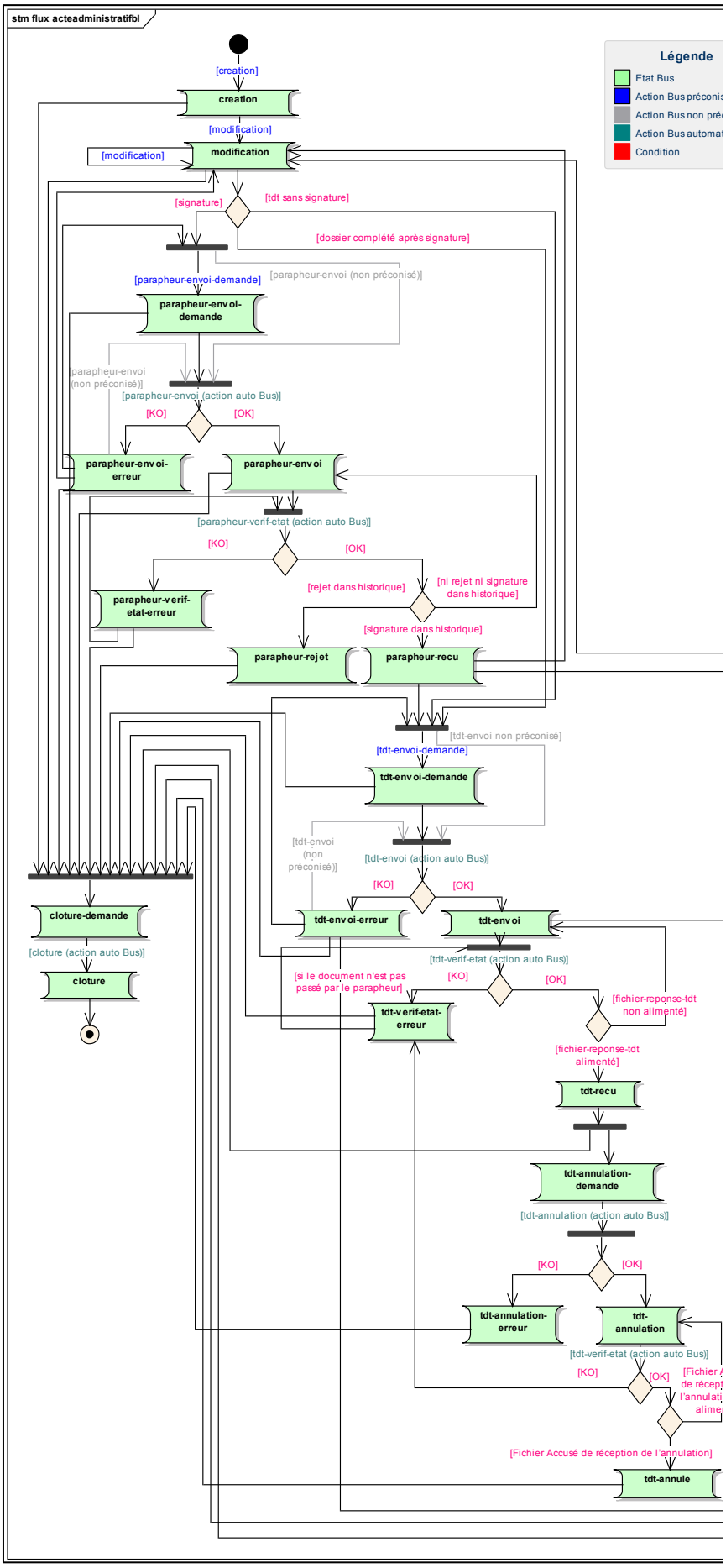


Figure 19 - Diagramme d'état bus BLES

## III.3 ARCHITECTURE BASEE BUS

Le Bus Applicatif est un ESB (Enterprise Service Bus) qui va permet d'adresser les besoins d'interopérabilité entre les applications Berger-Levrault, ainsi qu'avec les applications tierces qui voudront ou auxquelles nous voudrons nous interfacer. Il permettra d'implémenter de manière unique chaque nouvelle interface et aidera à la normalisation des échanges entre applications.

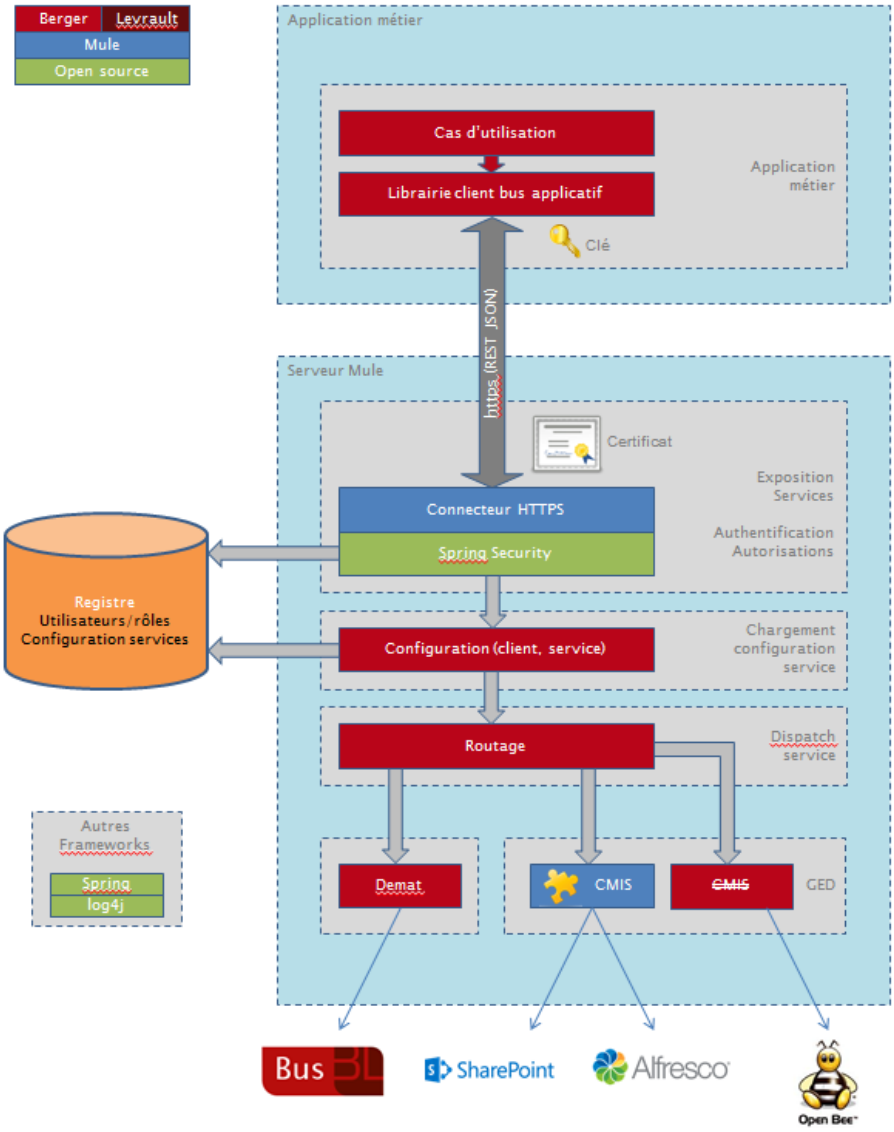


Figure 20-Architecture basée bus

Le Bus Applicatif s'articule autour de trois parties : les points d'entrée, les traitements et les demi-connecteurs de sortie.

Pour le moment, le seul point d'entrée est un Web service REST en HTTPS avec une authentification par login/mdp. Ceci implique que les échanges entre les applications et le Bus Applicatif sont cryptés.

Les traitements à l'intérieur du Bus Applicatif transforment le message entrant afin qu'il puisse être transmis aux services distants (comme une GED).

Enfin, les connecteurs de sortie transmettent le message au bon format vers le serveur paramétré dans le Bus Applicatif.

### III.3.1 BUS APPLICATIF BL

Le bus applicatif BL est un ESB développé avec le Framework Mule ESB. Mule est un Enterprise service bus (ESB) et un Framework d'intégration. La plate-forme est basée sur JAVA. L'architecture est évolutive, distribuable et peut gérer les interactions entre les systèmes existants, des applications clients, et presque tous les transports modernes et des protocoles.

#### III.3.1.1 MULE RUNTIME

Ce paragraphe présente les différents modes runtime pour Mule avec un tableau récapitulatif des avantages et inconvénients.

**Standalone**

La manière la plus simple de déployer Mule est d'installer un serveur en mode standalone.

Le serveur Mule peut être installé comme un service Windows (Windows OS) ou un daemon (Unix OS). Cela permet un démarrage/arrêt automatique du serveur Mule en même temps que l'OS.

Le serveur peut également être démarré manuellement en mode console.

Le mode de déploiement standalone s'appuie sur Java Service Wrapper* qui permet au serveur

- de s'exécuter comme un daemon Unix ou un service Windows,
- d'être notifié en cas de problème (crash) qui permet un redémarrage automatique du service.

Après décompression de l'archive, l'arborescence comprend les répertoires

- **./apps** : pour le déploiement des applications Mule, sous forme d'un fichier *.zip ou d'une arborescence de répertoires
- **./bin** : contenant les batches de lancement du serveur Mule pour les plateformes Unix et Windows,
- **./conf** : contenant les fichiers de configuration du serveur (log4j, Java Service Wrapper),
- **./domains/*** : contient des librairies qui peuvent être partagées par domaine, le domaine pouvant être définit pour chaque application dans le fichier mule-deploy.properties, par défaut domain=default. A noter que chaque application peut également déployer ses propres librairies dans son répertoire ./apps/{application}/lib *(cf. installation application Mule),*
- **./lib** : contenant les librairies qui sont chargées par un *classloader* hiérarchique qui permet au serveur et à chaque application d'avoir accès à des librairies et ressources propres ou partagées :
    - **./boot** : librairies de boot du serveur Mule,
    - **./endorsed** : librairies sérialisation XML, moteur XSLT,
    - **./mule** : librairies Mule ESB (modules, transports…),
    - **./opt** : libraires sur lesquelles s'appuie Mule (Spring, log4j, CXF…),
    - **./user** : contient les librairies externes transverses à toutes les applications,
    - *./lib/shared : déprécié depuis la version 3.5.0 et remplacé par ./domain.*
- **./logs** : contient les fichiers journaux.

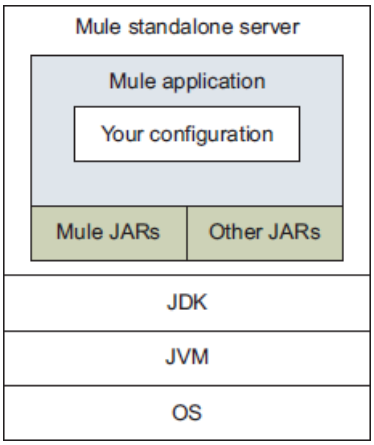


Figure 21 - Déploiement en mode standalone

Le *class loader* de Mule parcours les répertoires de lib dans l'ordre suivant

- alphabétique pour les répertoires de ./lib (./boot, ./endorsed, ./mule, ./opt, ./user),
- puis le répertoire ./apps/*/lib propre à l'application.
- prend la dernière surcharge d'une librairie dans l'ordre de ces répertoires.

Pour un connecteur que l'on développe ou une surcharge d'un connecteur Mule, déployer préférentiellement dans le répertoire ./lib/user.

**Dans une application web**

Mule peut être déployé comme intégré à une application web. Il suffit de rajouter le contexte Mule dans le contexte de l'application web. Mule bénéficie alors des fonctionnalités du container web (serveur HTTP, pool de connexion JDBC partagé entre applications...).

Le cycle de vie de Mule est implicitement lié au cycle de vie de l'application web. Mule est arrêté proprement en même temps que l'application web.

Cette pratique était courante pour les anciennes versions de Mule 2.x, mais a tendance à être abandonné au profit du mode de déploiement standalone.

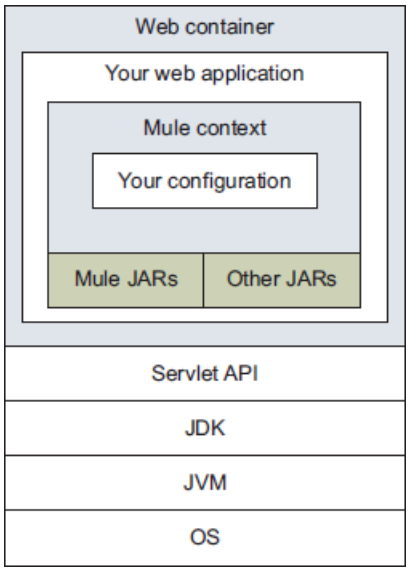


Figure 22 - Déploiement dans un container web

### Dans une application Java

Mule peut aussi être intégré dans une application Java existante.

Il peut ainsi servir de Framework de communication (protocoles de transport, tranformers, routeurs).

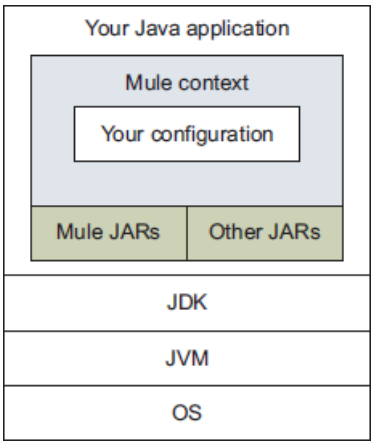


Figure 23 - Déploiement dans une application existante

### CloudHub

CloudHub est une IPAAS (Intégration Plateforme As A Service), plateforme d'intégration cloud proposée par Mule Soft.

**Comparaison**

Tableau 2- comparaison déploiement Mule

| | | |
|---|---|---|
| Standalone | • **installation simple et prêt à l'emploi**<br>• **déploiement/désinstallation à chaud des applications Mule**<br>• **robuste, extensible.** | • **partage de ressource entre applications** |
| Web Embedded | • partage de ressources entre applications Mule (pool de connexion JDBC par exemple) | • gestion des librairies Mule<br>• cycle de vie couplé au container |
| Java Embedded | • simple à déployer avec l'application<br>• Mule peut au choix monter le contexte Spring ou utiliser le contexte Spring de l'application | • gestion des librairies Mule<br>• cycle de vie couplé au container<br>• ne convient pas pour un mode serveur |
| Cloud hub | • scalabilité, haute disponibilité | • coût<br>• problème des applications nécessitant certains agréments légaux |

### III.3.1.2 PERSISTANCE

Pour garantir que tous les messages soient traités sans aucune perte, analysés en cas d'anomalie pour diagnostiquer le problème, ou tout simplement pour conserver l'historique, il faut persister les messages. Pour cela, plusieurs possibilités.

Le composant VM queue persisté utilise par défaut un stockage des messages sur le *file system* : en cas de crash, reprise des messages au démarrage. Cependant, quand l'instance arrêtée, il faut la redémarrer pour reprendre le traitement des messages. Si l'on souhaite l'utiliser dans un environnement type cluster (démarrage/arrêt d'instances en fonction de la charge), cela n'est pas possible avec l'*Object Store* par défaut.

Un broker de message externe (JMS, AMQP), comme Apacha ActiveMQ, HornetQ, pour garantir la persistance des messages, également le *retry delivery* et les *dead letter queue* quand le service est indisponible.

Une base de données, soit relationnelle, soit NoSql comme MongoDB. Cette base pourra également servir pour la persistance de la configuration, le monitoring/reporting du fonctionnement de l'ESB.

#### III.3.1.3 SECURITE

La sécurité est un point important et sensible dans le bus applicatif à tous les niveaux. Il faut mettre en place une gestion pour limiter les accès, la confidentialité de l'information et la fiabilité du système. Mule peut s'appuyer sur Spring Security ce qui permet d'adresser l'authentification et les autorisations.

**Authentification**

L'authentification sera rendue obligatoire pour utiliser une fonctionnalité du bus applicatif, qu'il s'agisse d'une authentification simple (type *user service* défini dans un fichier de configuration Spring), d'une authentification basée sur une gestion d'utilisateurs en base de données, d'une authentification qui se base sur un LDAP.

Il faudra probablement définir plusieurs authentifications en fonction des consommateurs (application Berger-Levrault, partenaire…) et des services consommés. Il peut même être envisagé un couplage authentification/client dans le cas d'un déploiement du bus en mode SAS.

Ceci ne permet pas de pouvoir définir plusieurs l'*authentication managers* dans une même application, notamment si l'on veut en fonction du service :

- une authentification simple basée sur une gestion d'utilisateurs définis dans un fichier ou une base de données,
- et s'appuyant sur un LDAP.

Il est nécessaire pour l'instant de définir tous les *authetication providers* dans un seul et unique l'*authentication manager (cf. POC securité)*.

**Mise en œuvre**

Utilisation au niveau des *inbound-endpoints* d'un *security filter*.

```
<http:inbound-endpoint exchange-pattern="request-response" ... >

      <spring-security:http-security-filter
securityProviders="securityProvider"

      realm="Merci de renseigner le login et le mot de passe afin d'accéder
au Bus BL"/>

    ...

</http:inbound-endpoint>
```

Si l'utilisateur n'est pas authentifié, une UnauthorizedException est levée.

**Autorisations**

Les endpoints exposés dans le bus applicatif peuvent être sécurisés et l'accès limité en fonction des autorisations de l'utilisateur.

Il est nécessaire de définir des rôles au niveau des *endpoints* du bus applicatif pour éviter qu'un utilisateur authentifié puisse avoir accès à toutes les fonctionnalités/services du bus

Utilisation au niveau *inbound-endpoint*s d'une *authorization filter* pour spécifier les roles ou *authorities* nécessaires

```
<http:inbound-endpoint exchange-pattern="request-response" ... >

    <spring-security:http-security-filter
securityProviders="securityProvider" ... />

    <spring-security:authorization-filter
requiredAuthorities="ROLE_GED"/>

    ...

</http:inbound-endpoint>
```

Si l'utilisateur n'a pas le rôle, une NotPermittedException est levée.

**Transport**

Il faut également sécuriser la couche de transport des messages pour garantir la confidentialité des données.

L'utilisation du protocole HTTPS permet de sécuriser les échanges en http. Cela introduit un peu plus de complexité dans la configuration de Mule et également pour les systèmes qui voudront consommer les services exposés.

Serveur HTTPS

Pour exposer du HTTPS, il faut utiliser un *keystore* dans lequel on importe un certificat ou on crée un *self-signed* certificat. Il suffit ensuite de définir dans un connecteur HTTPS le *tls-key-store* pour spécifier l'utilisation du *keystore.* Enfin, on fait référence à ce connecteur dans le *inbound endpoint*.

Le keystore est normalement crée lors de l'installation du serveur Mule en spécifiant des valeurs qui sont propres à l'installation (on peut éventuellement en fournir un par défaut).

```
<https:connector name="KEYSTORE" ... >
    <https:tls-key-store path="keystore.jks"
keyPassword="${keystore.keypassword}"
      storePassword="${keystore.storepassword}" />
</https:connector>
<https:inbound-endpoint connector-ref="KEYSTORE" ... />
    ...
</https:inbound-endpoint>
```

Client HTTPS

Pour consommer du HTTPS, il faut utiliser un *trust store* dans lequel importer le certificat demandé. Il suffit ensuite de définir dans un connecteur HTTPS le *tls-server* pour spécifier l'utilisation du *trust store*. Enfin, on fait référence à ce connecteur dans le *outbound* endpoint.

```
<https:connector name="TRUSTSTORE" ... >
    <https:tls-server path="truststore.jks"
storePassword="${truststore.storepassword}" />
</https:connector>


<https:outbound-endpoint connector-ref="TRUSTSTORE" ... />
    ...
</https: outbound-endpoint>
```

SOAP

Il est possible de sécuriser les échanges SOAP en utilisant WS-Security, une extension de SOAP qui définit des headers standard sécurisés en utilisant la librairie WSS4J. On peut également utiliser des certificats, pour plus d'informations, WS-Security CXF.

A noter qu'il est possible d'utiliser d'autres transports sécurisés par SSL/TLS : IMAPS, POP3S, SMTPS et XMPPS.

**Cryptage**

Il est également possible de crypter le contenu des messages pour augmenter la sécurite.

Plusieurs stratégies possibles :

PBE (Password Based Encryption) : cryptage par mot de passe mais qui suppose de partager l'information avec toutes les systèmes souhaitant crypter/décrypter les messages. Pourrait être utilisé en interne dans le bus et pour les applications Berger-Levrault mais ce n'est pas assez sécurisé pour l'externe.

PGP (Pretty Good Privacy) : utilisé pour crypter et décrypter des messages, mais également pour signer. Il est base sur l'utilisation des clés publique et privée. Mule utilise la librairie Cryptix. Ce module permet de gérer le cryptage (transformer) et peut être utilisé dans un *security filter* pour vérifier la signature du message.

Le cryptage des messages augmente encore la sécurité mais également complexifie l'interopérabilité avec le bus applicatif. Il faut disposer de l'algorithme de cryptage dans toutes les technologies (JAVA, .NET, PHP, VB6…). Je ne sais pas non plus si le bus peut imposer cette pratique dans le cas d'interopérabilité avec des partenaires extérieurs.

En revanche, comme le bus va probablement s'appuyer sur un *broker* de messages, il serait souhaitable que les messages envoyés dans les *queues* et *topics*

soient cryptés, car sinon, les informations apparaîtront en clair dans le *broker* de message.

Dans les cas d'utilisation où la sécurité exigée est maximale, le cryptage doit être envisagé.

*Mise en œuvre*

Pour un cryptage par mot de passe, il faut définir une stratégie de cryptage basée sur un mot de passe. Il

```
<security-manager>

    <password-encryption-strategy name="cryptingPBE"
password="${crypting.password}" />

</security-manager>
```

Pour le cryptage, il faut ajouter dans le flow un transformer :

```
...

  <encrypt-transformer strategy-ref="cryptingPBE" />

...
```

Pour le décryptage, il faut ajouter dans le flow un transformer :

```
...

  <decrypt-transformer strategy-ref="cryptingPBE" />

..
```

Tableau 3 - Comparatif sécurité bus BL

| Sécurité | | |
|---|---|---|
| Authentification | Basique s'appuyant sur une gestion d'utilisateurs/roles. | Actuellement, uniquement 2 utilisateurs *admin/ROLE_ADMIN* et *user/ROLE_USER* définis dans la configuration. Insuffisant |
| | LDAP | *Pas de cas d'utilisation pour l'instant* |
| Autorisations | Limiter l'accès aux *endpoints* en fonction des rôles de l'utilisateur. | Pas en place |
| Transport | HTTPS, *keystore*, *trust store* | Déjà en place dans le bus applicatif pour l'exposer les services et utiliser le Bus BL |
| Cryptage | Cryptage des messages. | *Pas de cas d'utilisation pour l'instant* |

La mise en œuvre dans les flows Mule des éléments sur l'authentification et les autorisations est simple, c'est la gestion des utilisateurs/rôles qui est plus complexe.

Actuellement, la configuration des services est gérée dans des fichiers au format json (clé/valeur). Faut-il passer sur une gestion en base de données ? Dans l'optique d'un environnement type *cluster* pou une configuration SAS avec un *load balancer* pour répartir la charge sur différentes instance Mule, éventuellement sur

des serveurs différents, le partage de la configuration serait simplifié par l'utilisation d'une base de données. Si on veut rester au format JSON, il est possible d'utiliser une base de données MongoDB, mais attention, plus de relationnel (éventuellement du *DBRef* pour référence).

D'une manière générale, cela rejoint la question de la gestion de la configuration (serveur, clients, utilisateurs, rôles, services…) qui sera traitée dans une autre partie.

### III.3.2 APPLICATIONS METIERS

Les applications métiers accèdent désormais à leurs services par le biais du bus applicatif. Afin de centraliser les appels une API est mise en place. Ainsi lors de l'implémentation du service dans le logiciel métier il ne reste qu'à appeler les méthodes du service nécessaire via l'api.

Les applications clientes de Berger Levrault doivent toute avoir accès à ce bus pour pouvoir les intégrer à l'architecture SOA telle qu'elle a été mise en place. C'est pourquoi il faut mettre en place une librairie JAVA qui sera également accessible aux autres applications métiers qui n'utilisent pas ce langage. C'est pourquoi il a été fait le choix de la librairie java qui peut être accessible à partir d'un patch pour les applications .NET et autres langages.

Cette librairie permet à la fois la connexion avec le bus applicatif en effectuant la gestion des certificats. Mais aussi la connexion à l'ensemble des services et sous services intégrés au sein de l'ESB. Les services correspondent à une classe spécifique. Les sous-services sont l'implémentation des appels à chaque fonctionnalité du service mise en place. Il est donc indispensable que la classe de service comprenne l'intégralité des sous-services. Une fonction correspond à un sous service qui sera appelé via la librairie sous le modèle d'un web service.

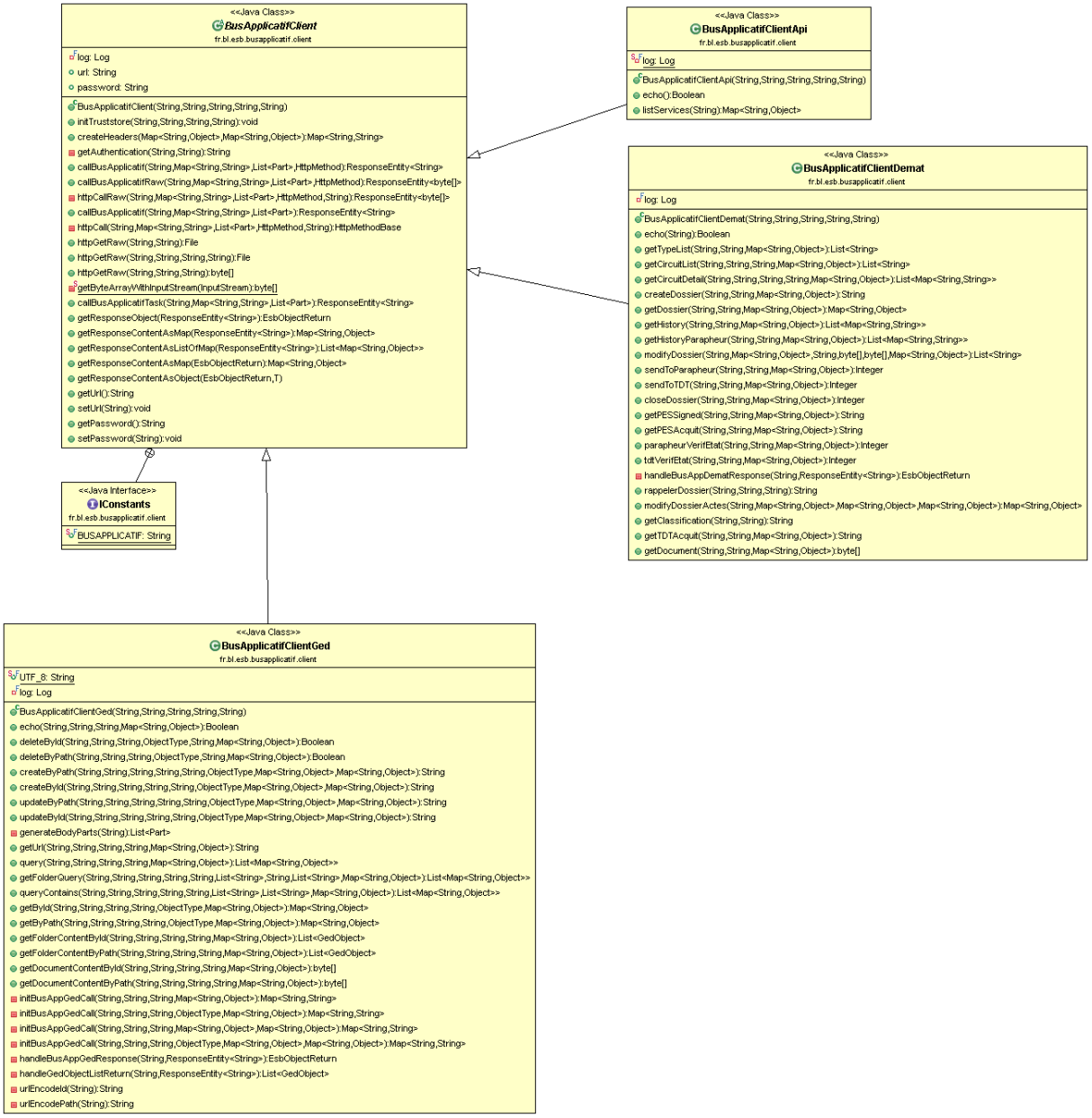


**Figure 24 - Diagramme de classe api client**

La classe principale *BusApplicatifClient* permet les appels au bus applicatif par les méthodes httpcall et httpGet pour la réponse. Les services sont dont implémentés dans une classe qui hérite de la première. Celle–ci expose l'intégralité des appels disponible pour ce service. Chaque ajout de service implique donc un développement particulier pour celui-ci.

### III.3.3 EXPOSITION DES WEB SERVICES

Il s'agit de définir comment exposer les web services du bus applicatif : REST ou SOAP. En théorie, ce n'est pas véritablement une problématique car il est possible d'exposer selon les 2 modes mais cela doublerait le travail en création/maintenance d'exposition.

**REST vs SOAP**

SOAP : Simple Object Access Protocol est un protocole au format XML. Il permet d'exposer des méthodes selon un contrat à découvrir défini dans un fichier WSDL (Web Service Description Language).

Les +

- contrat bien défini par le fichier wsdl,

Les -

- verbosité du XML (poids des messages),
- consommation complexe (client s'appuyant sur code proxy généré),

REST : Representational State Transfer est une architecture d'exposition « naturelle » de ressources/données basé sur le nom des ressources et les méthodes http (GET, POST…) pour les opérations de CRUD.

Les +

- Format JSON léger (pas de format obligatoire),
- consommation simple,

Les -

- pas de contrat bien défini *(quoi que : cf. RAML)*.

SOAP est un protocole qui est plus adapté dans le développement d'une application intégrée client/serveur et que l'on maîtrise les 2 côtés (client et serveur) en même temps. Il utilise des entités fortement typées ce qui peut être un inconvénient dans une architecture découplée.

REST est plus simple et léger à mettre en œuvre, est plus performant et scalable, peut bénéficier d'un cache. C'est pour ces raisons que son utilisation croît plus vite que pour SOAP, notamment pour les API exposées sur le web, alors que SOAP reste plutôt utilisé en développement d'application « entreprise ».

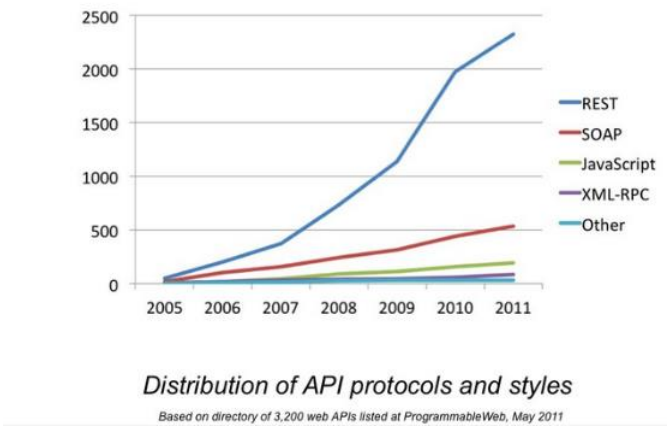


Figure 25 - Distribution des protocoles de distribution API 1

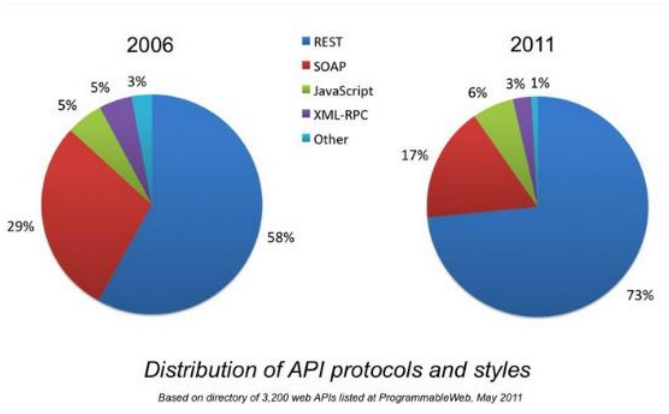


Figure 26 - Distribution des protocoles de distribution API 2

REST s'impose comme le choix par défaut pour l'exposition des web services. Cependant, il faut absolument pallier son défaut qui est l'absence de contrat. Pour cela, il faut absolument que l'API REST soit bien documentée et que celle-ci soit maintenue.

Il est possible que dans certains cas, nous soyons amenés à exposer des web services SOAP :

- En fonction des besoins ou des demandes,
- lorsqu'il s'agit d'exposer en proxy ou pseudo-proxy vers un web service SOAP,
- lorsque le contrat du web service nous est imposé *(cf. Interface Anticyclone Dic'T)*.

**Documentation REST**

Aussi naturelle que peut être la structuration d'une API REST, il n'en est pas moins nécessaire de fournir une documentation détaillée, à jour, avec des exemples à destinations des consommateurs de l'API. Rien de plus agaçant que de rencontrer des dysfonctionnements et incohérences lors de l'utilisation d'une API.

Pour remédier à ce problème, il existe différents outils pour documenter une API, dont les plus couramment utilisés :

- Swagger : permet de produire une description de l'interface API REST qui permet la découverte du service pour le développeur, les programmes et outils de test (peut se baser sur des annotations standards pour les web services),
- I/O Docs : idem,
- RAML : Restfull API Modelling Language est un language de description d'une Api REST. Il permet de décrire les ressources, méthodes, schémas en retour et exemples. Il permet :
    - dans AnyPoint Studio avec l'API Kit Router de générer le routage des appels entrants vers les flows correspondants (voir plus bas),

- d'exposer une console de test de l'API.
- de générer le code Java JAX-RS pour une implémentation Java de l'API REST.

### III.3.4 EXPOSITION REST AVEC MULE

**Pattern proxy**

Le premier cas d'utilisation est celui d'un proxy transparent avec simple redirection (changement de serveur et/ou d'url).

**REST Router Module**

Ce module est un message processor qui en fonction de template d'URI et des méthodes http va permettre de router vers un flow spécifique. Les paramètres de l'URI sont affectés à des varaibles de flow

```
<rest-router:router templateUri="/users/{userid}">
  <rest-router:get>
    <flow-ref name="GetUser" doc:name="GetUser" />
  </rest-router:get>
  <rest-router:delete>
    <flow-ref name="DeleteUser" doc:name="DeleteUser" />
  </rest-router:post>
</rest-router:router>
```

Ce module n'a plus évolué depuis 2 ans et est maintenant remplacé par l'API Kit router *(cf. ci-dessous)*, donc à éviter.

**Jersey REST Module**

Ce module Mule permet d'exposer un service REST JAX-RS contenant des méthodes annotées (*@Path, @Get|@Post|…, @PathParams…*) et peut s'appuyer sur un wrapper pour la retourner la réponse.

```
<jersey:resources doc:name="REST">
  <component class="fr.bl.esb.poc.rest.jaxrs.UsersService" />
  <jersey:exception-mapper
class="fr.bl.esb.poc.rest.common.BadURIExceptionMapper" />
</jersey:resources>
```

```
@Path("users")
public interface UsersService {
    /**
     * Create a user
     * @param entity
     *      e.g. { "user": { "id": 1, "name": "John DOE", "login": "jdoe" }
}
     */
    @POST
    @Consumes("application/json")
    @Produces({
        "application/json"
    })
    Users.PostUsersResponse postUsers(Object entity)
        throws Exception
    ;
public class PostUsersResponse
        extends fr.bl.esb.raml.test.support.ResponseWrapper
```

```
        {
            private PostUsersResponse(Response delegate) {
                super(delegate);
            }

            /**
             * @param entity
             *  e.g. { "response": { "status": "0", "error": "" } }
             */
            public static Users.PostUsersResponse jsonOK(Object entity) {
                Response.ResponseBuilder responseBuilder =
Response.status(200).header("Content-Type", "application/json");
                responseBuilder.entity(entity);
                return new Users.PostUsersResponse(responseBuilder.build());
            }
        }
```

Cette méthode permet d'implémenter l'API par du code Java.

Il est également possible en utilisant la fonctionnalité de Component Bindings de router les méthodes du service vers des flows, mais dans ce cas, mieux vaut utiliser l'API Kit Router.

### API Kit Router

L'API Kit est un outil Mule qui permet de générer, à partir d'un fichier RAML (RESTful Api Modeling Langage), le squelette des flows Mule pour implémenter l'API. Pour chaque ressource/méthode, un flow est créé. Il crée également une stratégie d'exception par défaut (resource not found, invalid method…).

```
    <apikit:config name="users-config" raml="users.raml" consoleEnabled="true"
consolePath="console" doc:name="Router"/>

    <apikit:mapping-exception-strategy name="users-apiKitGlobalExceptionMapping"
doc:name="Mapping Exception Strategy">

            <apikit:mapping statusCode="404">

                <apikit:exception
value="org.mule.module.apikit.exception.NotFoundException" />

                <set-property propertyName="Content-Type"
value="application/json" doc:name="Property"/>

                <set-payload value="{ &quot;message&quot;: &quot;Resource not
found&quot; }" doc:name="Set Payload"/>

            </apikit:mapping>

            <apikit:mapping statusCode="405">

                <apikit:exception
value="org.mule.module.apikit.exception.MethodNotAllowedException" />

                <set-property propertyName="Content-Type"
value="application/json" doc:name="Property"/>

                <set-payload value="{ &quot;message&quot;: &quot;Method not
allowed&quot; }" doc:name="Set Payload"/>

            </apikit:mapping>

            <apikit:mapping statusCode="415">

                <apikit:exception
value="org.mule.module.apikit.exception.UnsupportedMediaTypeException" />

                <set-property propertyName="Content-Type"
value="application/json" doc:name="Property"/>

                <set-payload value="{ &quot;message&quot;: &quot;Unsupported
media type&quot; }" doc:name="Set Payload"/>

            </apikit:mapping>

            <apikit:mapping statusCode="406">
```

```
                <apikit:exception
value="org.mule.module.apikit.exception.NotAcceptableException" />

                <set-property propertyName="Content-Type"
value="application/json" doc:name="Property"/>

                <set-payload value="{ &quot;message&quot;: &quot;Not
acceptable&quot; }" doc:name="Set Payload"/>

            </apikit:mapping>

            <apikit:mapping statusCode="400">

                <apikit:exception
value="org.mule.module.apikit.exception.BadRequestException" />

                <set-property propertyName="Content-Type"
value="application/json" doc:name="Property"/>

                <set-payload value="{ &quot;message&quot;: &quot;Bad
request&quot; }" doc:name="Set Payload"/>

            </apikit:mapping>

        </apikit:mapping-exception-strategy>

        <flow name="users-main" doc:name="users-main">

            <http:inbound-endpoint address="http://localhost:8081/api"
doc:name="HTTP"/>

            <apikit:router config-ref="users-config" doc:name="APIkit Router"/>

            <exception-strategy ref="users-apiKitGlobalExceptionMapping"
doc:name="Reference Exception Strategy"/>

        </flow>

        <flow name="put:/users/{userid}:users-config"
doc:name="put:/users/{userid}:users-config">

            <set-payload value="{&#xA;  &quot;response&quot;: {&#xA;
&quot;status&quot;: &quot;0&quot;,&#xA;    &quot;error&quot;: &quot;&quot;&#xA;  }
&#xA;}" doc:name="Set Payload"/>

        </flow>
```

```
        <flow name="delete:/users/{userid}:users-config"
doc:name="delete:/users/{userid}:users-config">

            <set-payload value="{&#xA;  &quot;response&quot;: {&#xA;
&quot;status&quot;: &quot;0&quot;,&#xA;    &quot;error&quot;: &quot;&quot;&#xA;  }
&#xA;}" doc:name="Set Payload"/>

        </flow>

        <flow name="get:/users/{userid}:users-config"
doc:name="get:/users/{userid}:users-config">

            <set-payload value="{&#xA;  &quot;user&quot;: {&#xA;
&quot;id&quot;: 1,&#xA;    &quot;name&quot;: &quot;John DOE&quot;,&#xA;
&quot;login&quot;: &quot;jdoe&quot;&#xA;  }          &#xA;}" doc:name="Set
Payload"/>

        </flow>

        <flow name="post:/users:users-config" doc:name="post:/users:users-
config">

            <set-payload value="{&#xA;  &quot;response&quot;: {&#xA;
&quot;status&quot;: &quot;0&quot;,&#xA;    &quot;error&quot;: &quot;&quot;&#xA;  }
&#xA;}" doc:name="Set Payload"/>

        </flow>
```

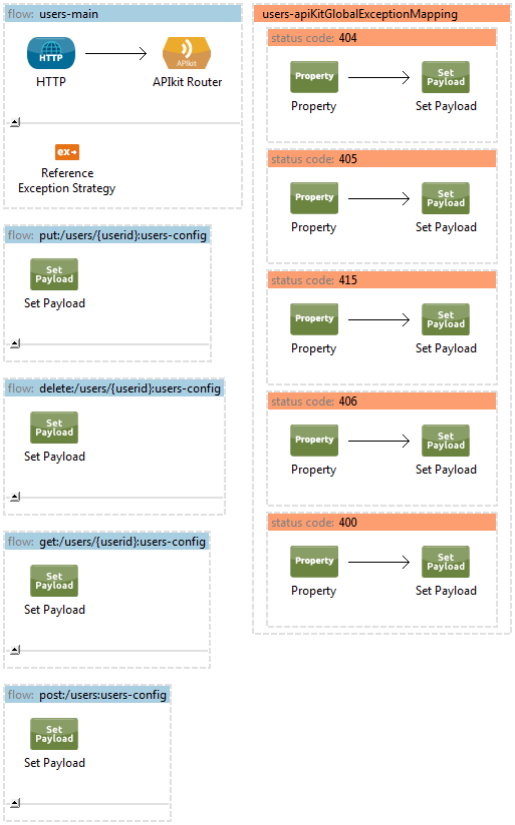


Figure 27 – Représentation graphique flows Mule

Il faut ensuite implémenter chaque resource/méthode par le flow Mule correspondant.

A noter qu'il permet également d'exposer une console web (dans eclipse ou depuis un navigateur) qui permet de tester l'API déployée sur le serveur.

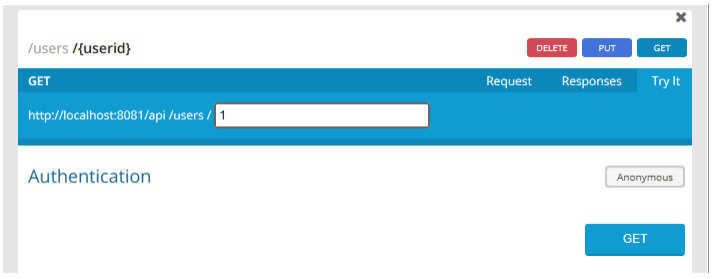


Figure 28 - Console web de test

Le flux utilise le langage propre à Mule ESB qui est le MEL (Mule Exchange Langage). Celui-ci  permet de gérer de façon graphique l'ensemble des éléments nécessaire à la gestion des flux.

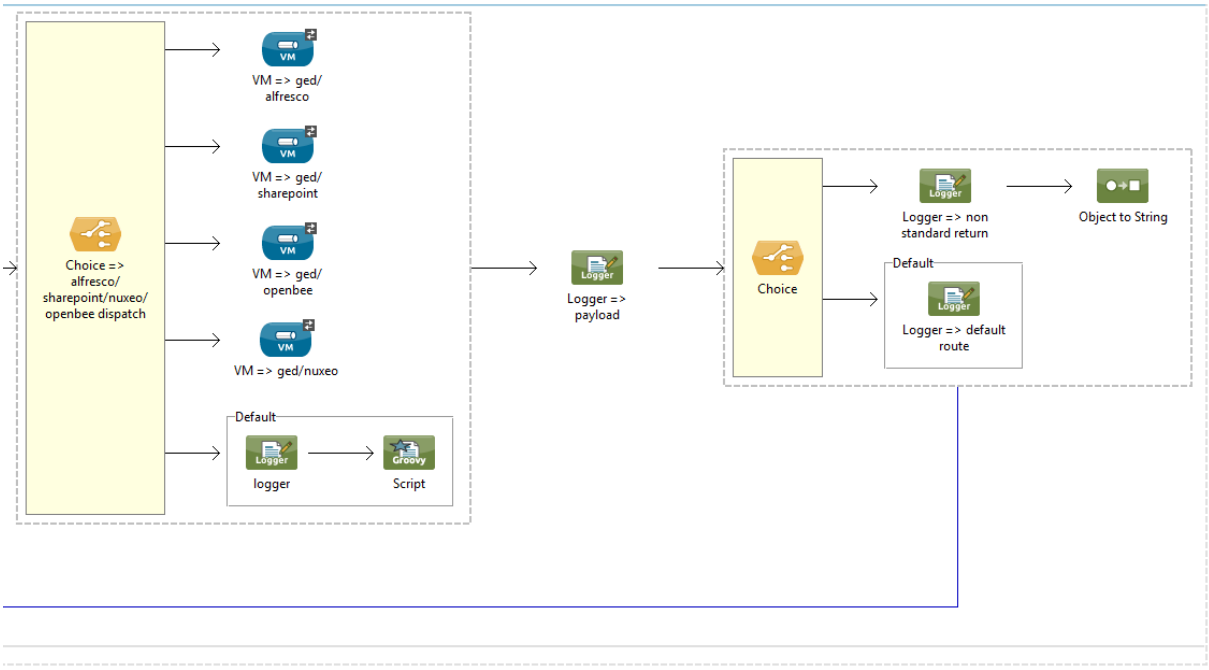


Figure 29 - exemple de flux

Nous pouvons voir sur la représentation graphique de flux qu'une redirection s'effectue au niveau du service de GED général vers le sous-service spécifique, ici alfresco. Les appels sont donc centralisé à l'entrée du bus puis spécifique au niveau du connecteur de service.

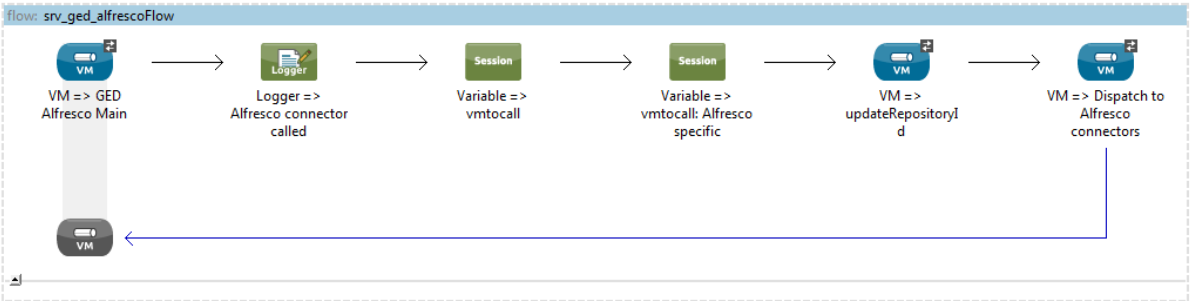


Figure 30 - redirection de sous services

### III.3.4.1 SERVICES DE GED

Toutes les communications vers le Bus Applicatif respectent les principes REST, avec une communication sécurisée par HTTPS. Certains entêtes HTTP sont nécessaires pour le bon fonctionnement des connecteurs, ainsi que du Bus Applicatif :

- **`clientId`**, est obligatoire, doit contenir la clé correspondant au client appelant. Permet de récupérer dans le paramétrage du Bus Applicatif la configuration spécifique du client
- **`Authorization`**, est obligatoire, contient l'authentification HTTP basique pour l'accès au Bus Applicatif
- **`serviceId`**, non obligatoire, contient l'identifiant de service au cas où le client a plusieurs services configurés pour un connecteur donné
- **`serviceData`**, non obligatoire, contient les données au format JSON à destination du connecteur, le format dépend du connecteur, et du service demandé. Le détail pour le connecteur GED est décrit dans ce document

**Retours du Bus Applicatif**

Les retours du Bus Applicatif sont standards, à savoir que le retour peut être soit un flux brut de données dans le cas d'un téléchargement par exemple, soit un retour au format JSON. Dans tous les cas, une exception levée par le Bus ou le connecteur renvoie une erreur 500.

Le format de retour JSON respecte le schéma suivant :

- **`status`**, entier, contient 0 si tout est OK, différent de 0 si une erreur est survenue
- **`errorMessage`**, chaîne de caractère, contient null si status=0, le message d'erreur sinon

- **`errorType`**, chaîne de caractère, contient null si status=0, le type de message sinon, à savoir TECHNIQUE, BUS APPLICATIF ou CONNECTEUR
- **`content`**, chaîne de caractère, contient null si status !=0, le retour du service appelé sinon. Le contenu dépend du service appelé.

Les sous-services sont en annexe.

# IV REALISATIONS

Afin de mettre en pratique cette solution, le logiciel Actes office a été intégré dans l'architecture qui vient d'être décrite. Afin d'effectuer cette intégration il a été nécessaire de mettre en place le service de dématérialisation. Ce service n'étant pas encore conceptualisé ou développement, mon projet était de le concevoir.

Les connecteurs sont donc la première étape de l'intégration. Puis le développement dans l'application en elle-même, pour finir par une phase de test afin de vérifier  l'intégrité et la corrélation avec les besoins.

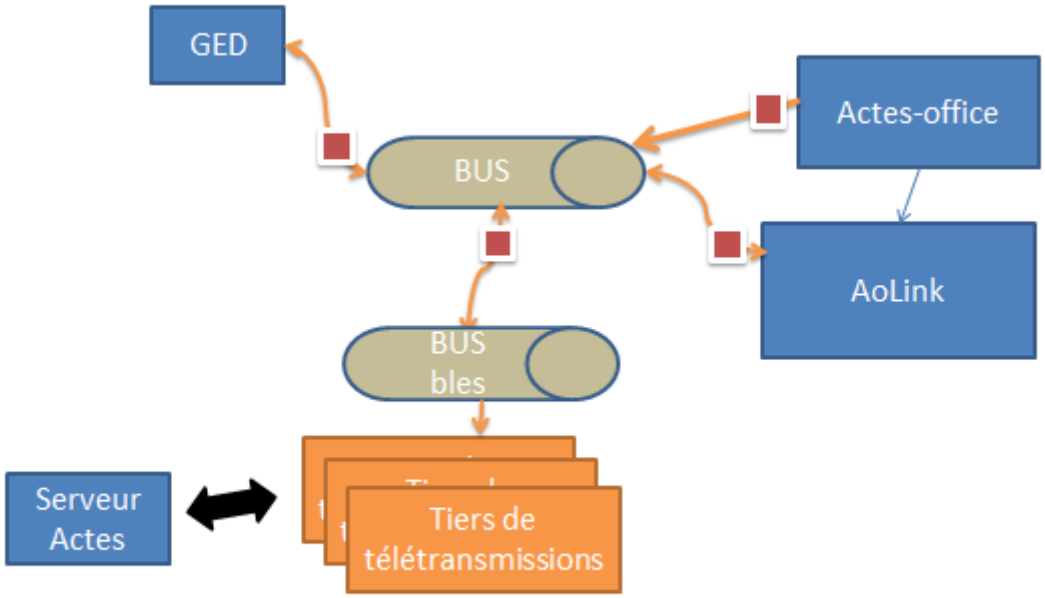


**Figure 31 - Architecture finale d'Acte office**

## IV.1 CONNECTEURS

L'intégration d'Actes office dans l'architecture a nécessité la mise en place du service de dématérialisation. Cette étape nécessite le développement des connecteurs à la fois coté client, que l'on nomme l'Api bus, et coté bus, le service en lui-même.

### IV.1.1 MISE EN PLACE DE L'API BUS

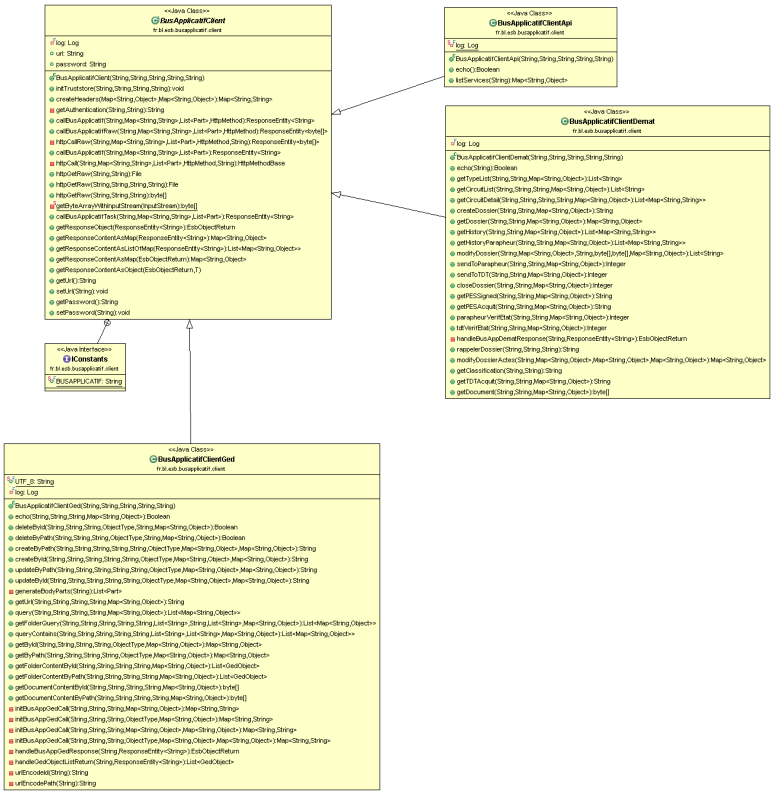


**Figure 32 - API client**

La librairie comme vu dans le chapitre sur l'implémentation de l'API doit répondre à tous les besoins de services liés au bus BLES dans notre cas. Il existe aussi des besoins particuliers du fait du retour venant des Tiers de télétransmissions. En effet, le retour du Tdt se fait au format XML, ce qui implique de créer le document en partant d'un flux JSON. Des méthodes de gestion des documents PDF, des documents et du flux JSON sont donc nécessaires.

L'api client nécessite aussi la mise en place de tests unitaires afin de vérifier que l'ensemble des appels envois les informations correspondantes. Cela engendre une problématique au niveau des retours des tdts qui ne sont pas immédiats.

Le rappel d'un acte qui vient d'être envoyé pose problème puisque tant que le bus BLES n'a pas reçu d'accusé de réception, il ne peut pas mettre en action, l'état de retour du document. Ainsi afin de vérifier l'intégrité de l'ensemble des états il st nécessaire de simuler certaines réactions du service.

### IV.1.2 CONNECTEUR SERVICE DEMATERIALISATION

Le connecteur du service de dématérialisation répond donc à l'ensemble des actions présentées dans la figure 19. Il doit répondre à chacune des actions qui permettent à l'acte de passer par tous les états dont il dépend.

Mais il doit aussi être généraliste pour qu'il puisse permettre de laisser passer dans le flow à la fois des éléments textes, tout comme des documents de type PDF. Les flux entrant sont simple, puisqu'ils correspondent aux paramètres d'entrée des web services du bus BLES. La transformation des flux sortant et quant à elle plus complexe.

Par défaut le flux va modifier les réponses afin qu'elles soient enveloppées dans un flux JSON. Il se pose alors le problème des flux sortant qui correspondent à un document de type PDF. En effet, ceux-ci contrairement aux documents XML ne peuvent pas être transformés en flux texte, ils seraient alors illisible. Il faut donc mettre en place un système de réponse http spécifique aux besoins d'un document PDF. Une réponse http avec un header comprenant les informations du flux PDF a été développée.

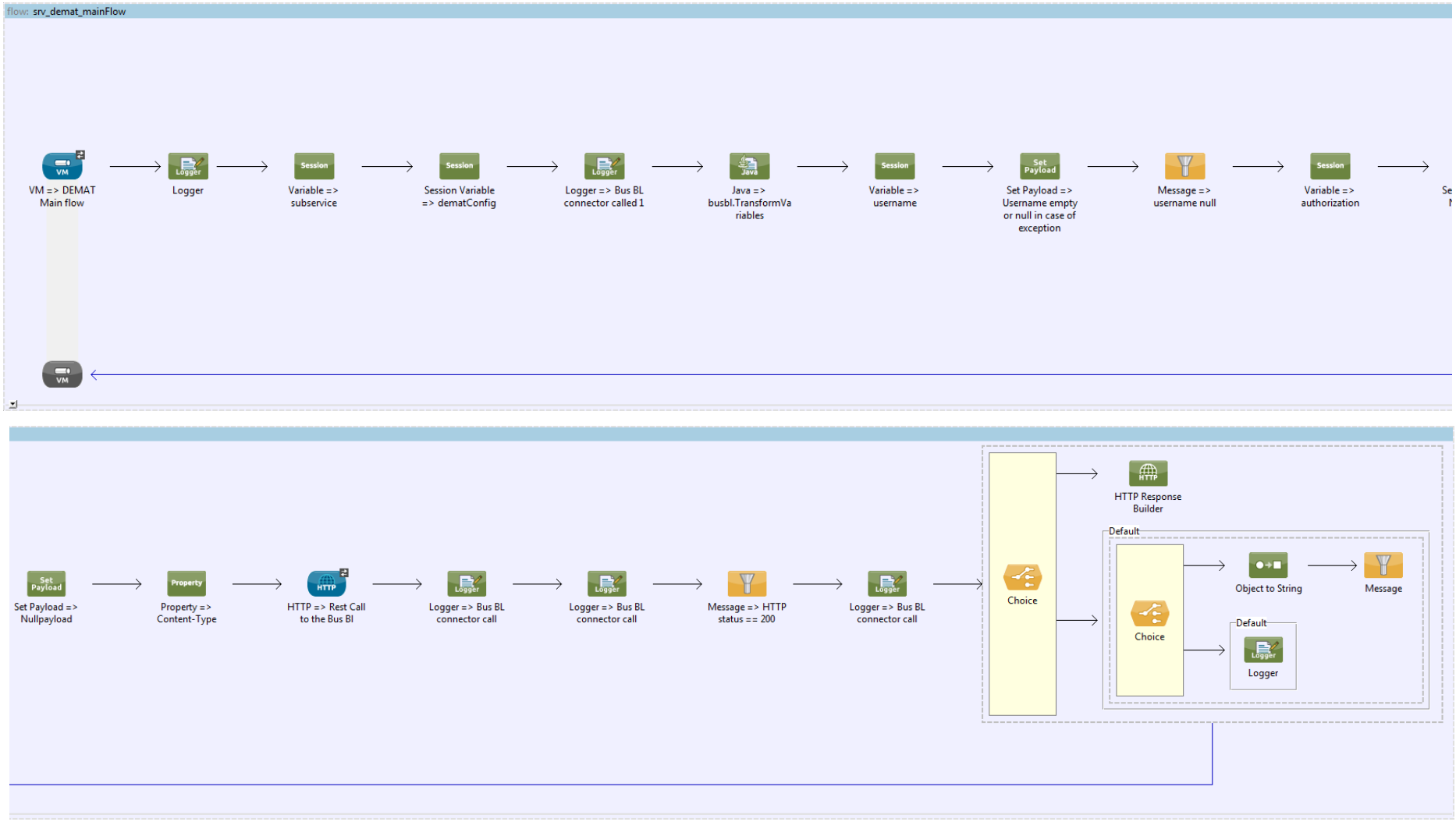


Figure 33 - Flux du service de dématérialisation

# IV.2 PROJET

## IV.2.1 INTEGRATION D'ACTES OFFICE DANS L'ARCHITECTURE

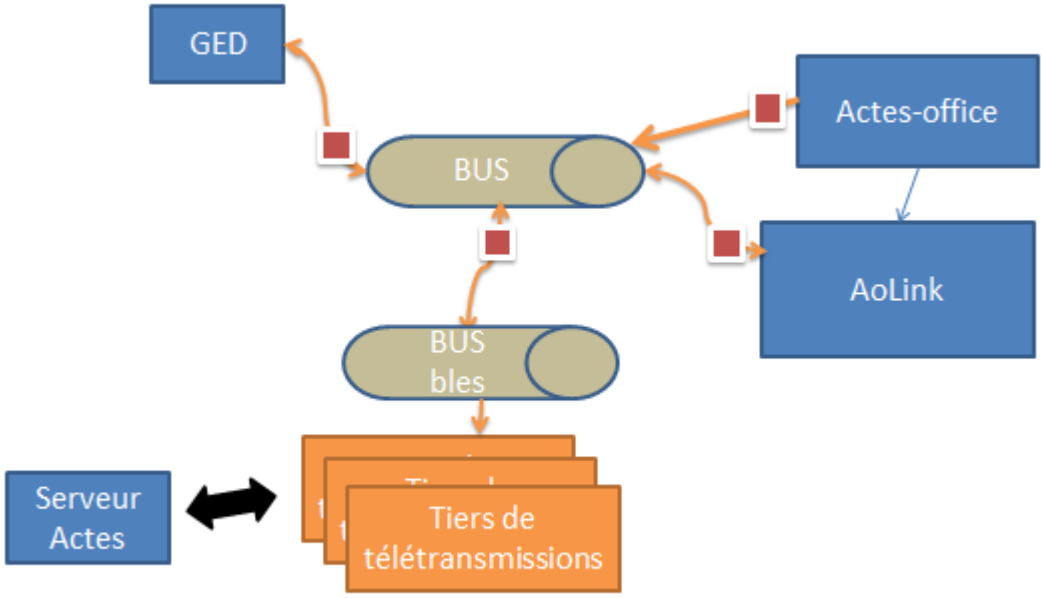


Figure 34- Architecture Actes-office finale

L'intégration d'AoLink dans cette architecture pose déjà le problème de la perte de compétences, mais aussi celui de la rétrocompatibilité.

Lorsque je suis arrivée sur le projet, l'équipe de développement d'Actes office avait subi une perte de compétence au niveau du module AoLink. En effet, un besoin été exprimé de le modifier, mais personne n'avait les compétences et les connaissances sur ce logiciel. La première étape était donc d'analyser son fonctionnement afin d'effectuer une documentation de faisabilité dans son intégration dans l'architecture SOA. Cette analyse a permis à l'aide du reverse engineering de définir l'architecture d'implémentation des tiers au sein du logiciel.

Le besoin de rétrocompatibilité a aussi été un élément majeur d'analyse de l'implémentation des tiers de télétransmissions. Cette architecture fonctionne sur le design pattern de l'interface. Ainsi chaque Tiers possède ses classes spécifiques mais peut les implémenter dans des fonctions spécifiques.

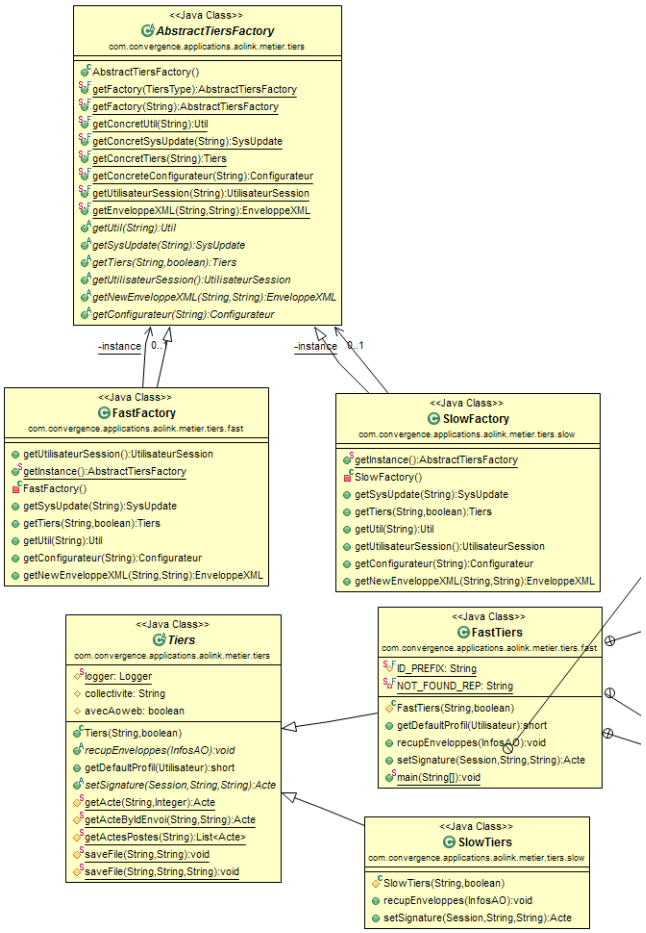


**Figure 35 - implémentation Tdt AoLink**

L'élément le plus difficile à mettre en place a été d'insérer un tiers de télétransmission qui ne possède pas les éléments générique d'un Tdt, puisqu'il s'agit d'un service de l'architecture et non pas d'un Tdt en lui-même.

### IV.2.2 IMPLEMENTATION DE LA GED

L'implémentation de la GED dans Actes office répond au besoin de gestion des documents. Les utilisateurs ayant le choix de plusieurs GED, il est donc nécessaire d'intégrer Actes-office dans cette architecture afin de ne pas se retrouver avec la même problématique que pour la télétransmission des Actes dans AoLink.

Les connecteurs GED pour le bus ont déjà été développé, il s'agit donc d'intégrer cette architecture et de créer les IHM pour le client.

#### IV.2.2.1 ENREGISTREMENT DES DOCUMENTS

Une des principales fonctionnalités dans la GED est l'envoi des actes à celle-ci. Des Métadonnées peuvent être attachées au document lors de l'envoi (cf. paragraphe métadonnées).Avant toute chose il faut définir quel type de documents sont à envoyer à la GED.

##### Document à envoyer

Il apparait que le seul document intéressant à envoyer soit le **document final**.

Le document final est le document de l'acte après le contrôle de légalité : soit l'acte télétransmit avec le tampon virtuel si l'on passe par AoLink soit un scanne de l'acte après validation physique.

Le bordereau d'acquittement récupéré par AoLink depuis le TDT ou scanné peut également être lié au document final en pièce jointe.

##### Cas d'utilisation

L'envoi à la GED peut s'effectuer soit de façon automatique, soit de façon manuelle. Ce choix s'effectue sur la page de paramétrage.

**Scénario envoi automatique :**

Il doit être possible de définir le moment où un document doit être envoyé dans la GED lorsque ce dernier est automatique. Ce déclencheur est paramétrable.

- Par défaut lorsque l'utilisateur génère le document final.
- Régulièrement (exemple : tous les jours à 5h du matin).

**Scénario envoi manuel :**

L'envoi manuelle nécessite de mettre en place deux scénarios : un envoi actes par actes ou en lot. L'envoi par lot peut être utilisé lors de la première connexion à la GED pour envoyer tous les documents finaux déjà générés.

- **Envoi en lot** : permet d'envoyer l'ensemble des documents sélectionnés à la GED. IL est nécessaire d'ajouter une page accessible depuis un onglet « GED » du menu « exporter ».


Figure 36 - Page d'export GED

Cette page affiche uniquement les délibérations, arrêtés ou décisions qui possèdent un document final et ne sont pas déjà dans la GED.

L'envoi s'effectue en cochant les check box pour les actes que l'on veut envoyer puis en cliquant sur le bouton « Exporter à la GED ».

La loupe permet de visualiser le document final.

- **Envoi pour un acte** : pour envoyer un acte spécifique depuis la fiche de l'acte accessible depuis le menu « Préparer ».

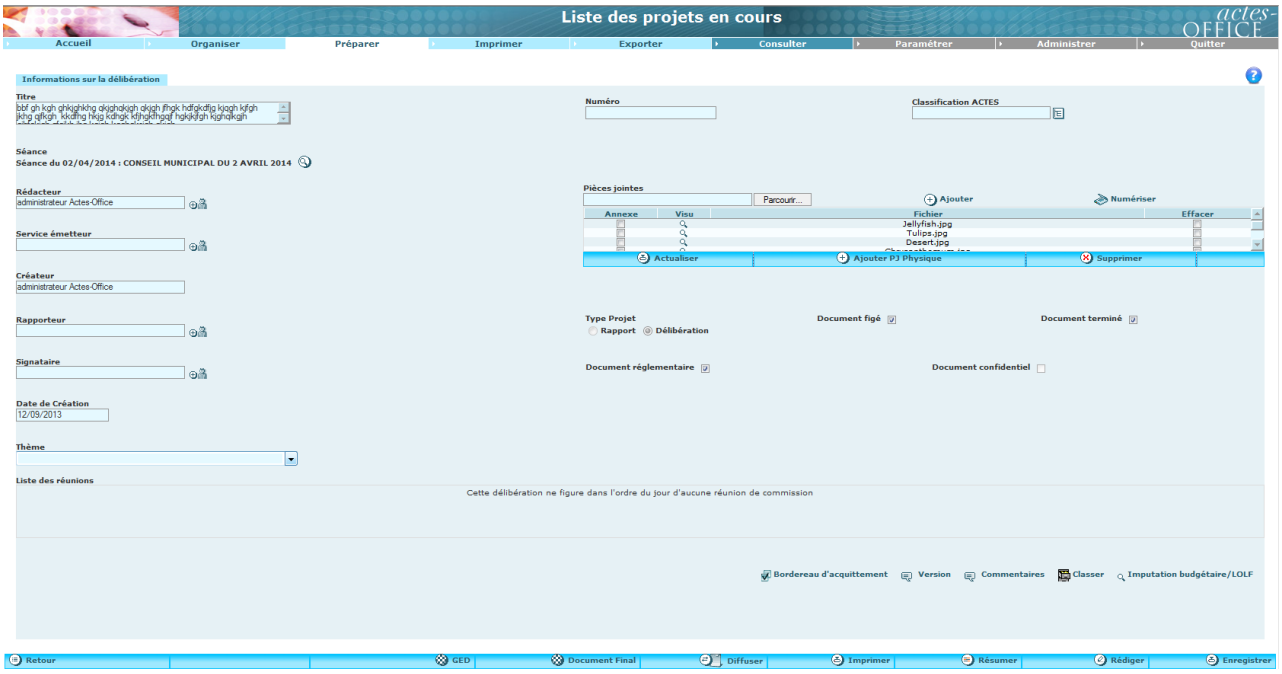


Figure 37 - Export GED

Si un document final existe le bouton « GED » s'affiche sur la barre d'outils en bas de la page.

Un clic sur le bouton GED affiche la boite de dialogue suivante :

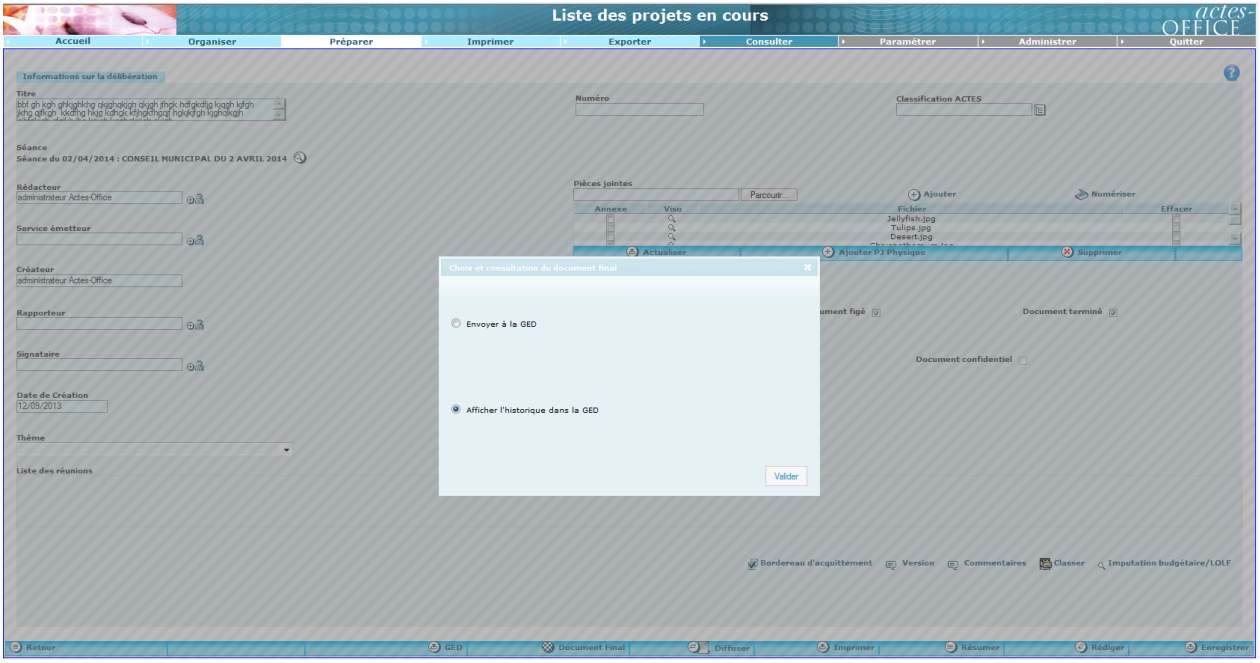


Figure 38 - historique GED

Dans la boite de dialogue, deux actions sont sélectionnables:

- *Envoyer à la GED* : en appuyant sur « valider » le document est envoyé à la GED. Cette action ne s'affiche que si le document n'est pas dans la GED.
- *Afficher l'historique dans la GED* : permet d'afficher l'historique de ce document dans la GED. Ce radio bouton n'est accessible que si le document est déjà dans la GED.

#### VII.2.2.1 MODIFICATION DES DOCUMENTS

L'enregistrement des modifications du document dans la GED diffère selon que l'utilisation est automatique ou manuelle.

**Paramétrage automatique :**

Toute modification du document final après le premier envoi, doit être envoyée à la GED de façon automatique, l'action d'enregistrer la modification entraine un envoi automatique à la GED.

**Paramétrage manuel :**

L'actualisation du document en GED s'effectue depuis la page d'historique de la GED via un bouton « mise à jour du document».

#### VII.2.2.2 HISTORIQUE DES DOCUMENTS

Il doit être possible de visualiser l'historique ainsi que la version du document qui est présente dans la GED. Cette fonctionnalité nécessite la mise en place d'une nouvelle page.

Cette page d'historique est accessible depuis la page « préparer » en visualisant l'acte (depuis la dialogue box page 4).

Un icone est à mettre en place dans l'onglet préparer afin d'afficher si un document est dans la GED.

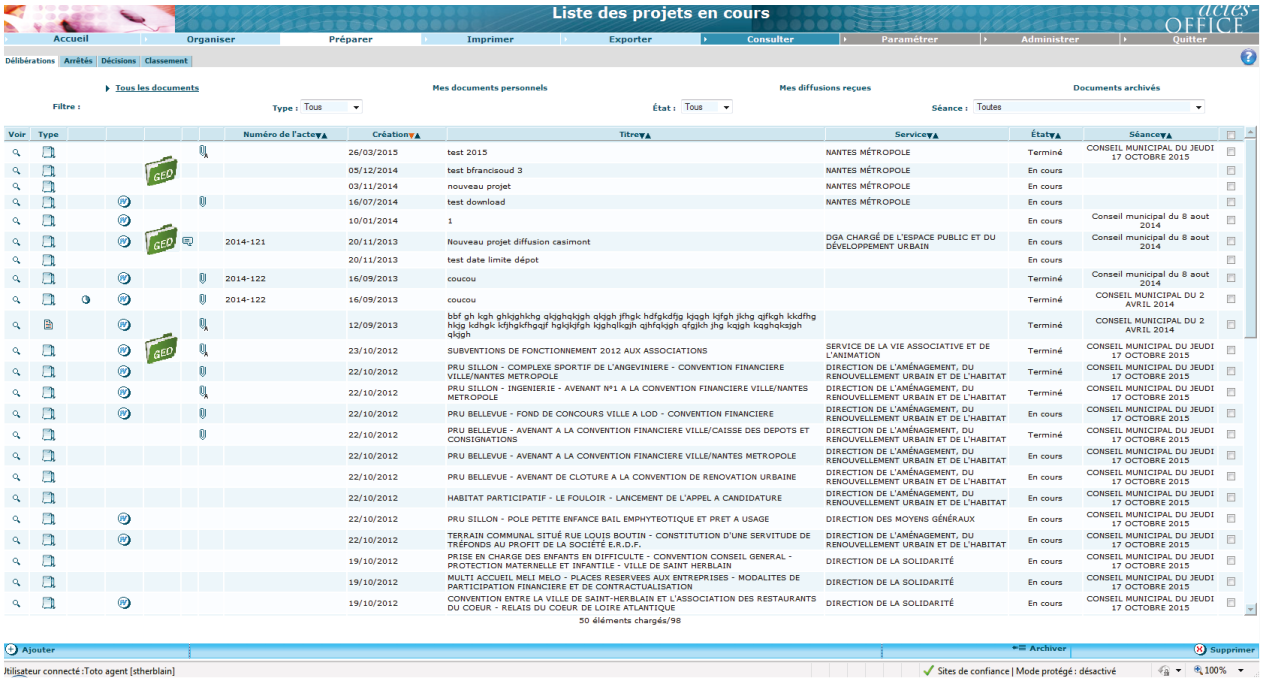


**Figure 39 - Page permettant de voir si un document est dans la GED**

La page d'historique doit comprendre :

- Le nom du document
- La date et l'heure de création
- L'utilisateur Actes Office pour l'envoi
- Les différentes modifications du document :
    - L'auteur
    - La date de la modification
    - L'heure de la modification
    - La version du document
    - Un lien pour télécharger la version du document.
- Les métadonnées associées au document.
- Un bouton permettant de mettre à jour le document dans la GED

### VII.2.2.3 PARAMETRAGE

Il doit être possible de paramétrer la GED depuis Actes Office afin de choisir si l'envoi à la GED des documents est automatique ou manuel.

Le paramétrage s'effectue à partir de la page Administrer, onglet Paramétrage et création d'un quatrième volet à gauche : « GED ».

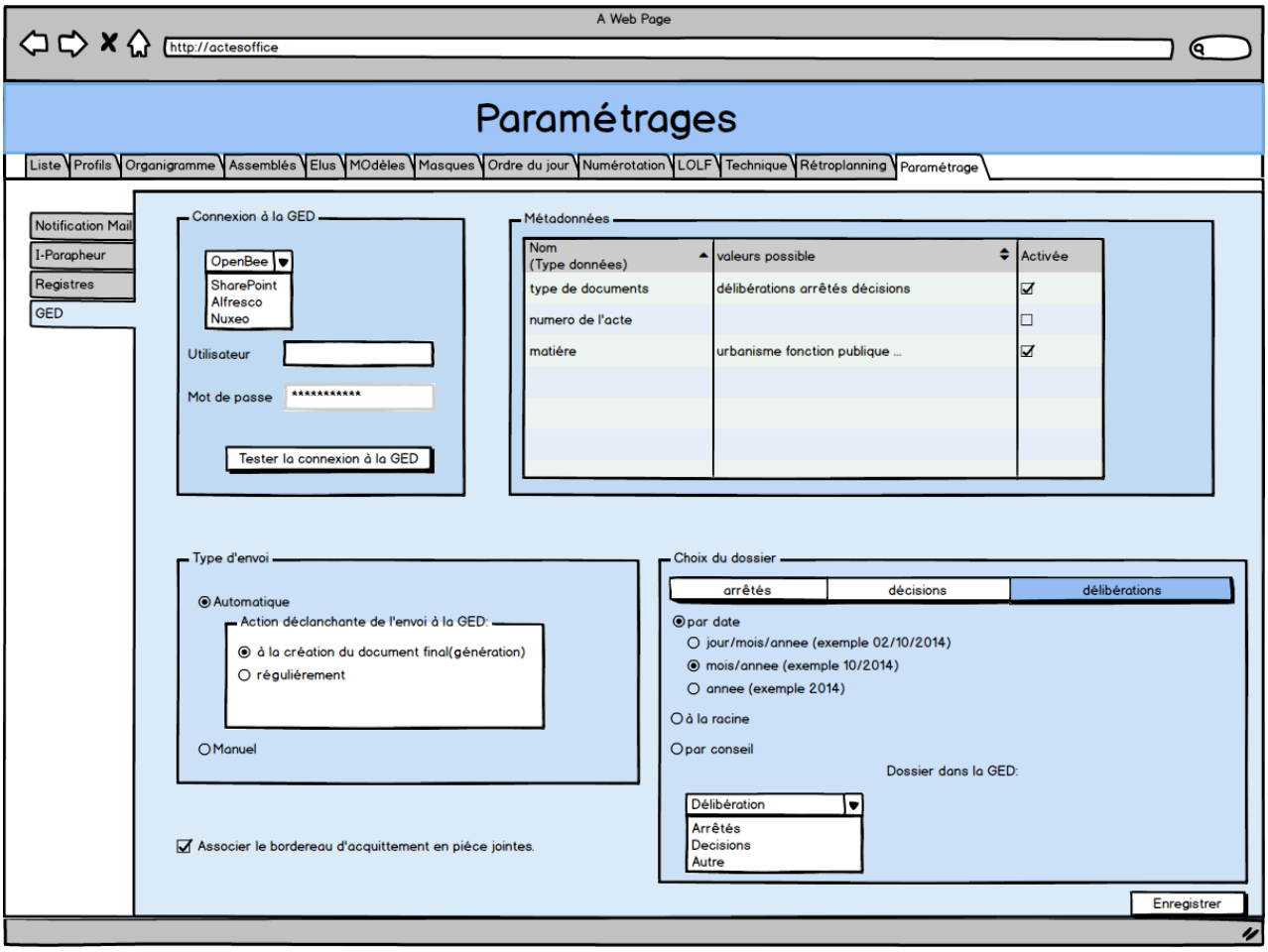


Figure 40 - Page de paramétrage de la GED

**Choix de la GED :**

L'architecture mise en place avec le bus applicatif permet une connexion à Alfresco, SharePoint, Nuxeo et Open Bee. Actes Office possédant déjà une connexion au bus il est possible d'utiliser une de ces 4 GED. L'utilisateur choisi donc la GED qu'il souhaite utiliser en renseigne les informations de connexion nécessaire.

Il est possible de tester la connexion avec la GED.

**Métadonnées :**

Il est possible de paramétrer les métadonnées que l'on veut associer aux documents au moment de l'envoi à la GED. Il est aussi possible de les activer ou non.

**Type d'envoi :**

La sélection du type d'envoi que l'on veut.Les informations à fournir varient selon le paramétre d'envoi.

- **Automatique :**

  Sélection de l'évènement déclencheur.

- **Manuel :**

  Il n'y a pas d'autres paramétrages si cette option est choisie.

**Choix du dossier :**

Permet de paramétrer où les documents sont envoyés dans la GED. Ce paramétrage s'effectue en deux étapes : le choix du dossier dans la GED et si des sous-dossiers ont besoins d'être crées.

Le choix s'effectue pour les arrêtés, décisions et délibérations. Par défaut le dossier utilisé et celui correspondant à la métadonnée associée au type de document. Il est cependant possible d'ajouter de choisir un autre dossier.

Il est aussi possible de choisir si un sous-dossier doit être crée pour ranger le document. Il existe plusieurs possibilités selon le type d'actes :

- Arrêtés/délibération :
  - *Par date* : les documents sont classés selon la date de l'acte. Par année, mois ou jour (quelques exemples : 2015, 05/2015, 07/2015…)
  - *A la racine* : depuis le dossier choisi.
- Délibérations :

- *Par date*
- Par conseil : le nom du dossier correspond au conseil au cours duquel la délibération à eux lieu
- *A la racine*

Ces sous-dossiers sont généré automatiquement (par exemple si le classement s'effectue par date (mois/années), le premier du mois le dossier est créé.

**Associer le bordereau d'acquittement :**

Il est possible de choisir si les bordereaux d'acquittement seront associés aux documents lors de l'envoi à la GED. Si cette option est choisie le bordereau sera en pièce jointe du document final.

**Les métadonnées**

Les métadonnées permettront le « rangement » automatique des documents dans la GED. Ces métadonnées peuvent être paramétrées afin de s'approcher des besoins spécifiques de la collectivité. Leur utilisation est optionnelle et dépendra des besoins.

Tableau 4 - Métadonnées GED

| Type | Valeurs | |
|---|---|---|
| Nature du document | Délibération, Arrêtés, Décisions | obligatoire |
| Titre de l'acte | | optionnelle |
| Thème (libelle) | Affaires culturelles, bâtiments, finances… | optionnelle |
| Date de création | 02/05/2014 | optionnelle |
| Date de l'acte | 02/12/2014 | optionnelle |
| conseil | Conseil municipal du 202/10/2014,… | optionnelle |
| commission | Commission affaires générales,… | optionnelle |

| | | |
|---|---|---|
| Séance | Séance du 08/02/2015, conseil municipal du 02/02/2015,… | optionnelle |
| Numéro de l'acte | 254-565, 565- | optionnelle |
| Classification ACTES | Commande publique, urbanisme, domaine et patrimoine… | optionnelle |
| rédacteur | Agent toto … | optionnelle |
| Service émetteur | Chargé des ressources, chargé de la vie sociale | optionnelle |

## Utilisateurs

La gestion des droits utilisateurs est différente selon que l'on parle de l'utilisateur de la GED ou de celui d'Actes Office.

**Utilisateur GED :**

Afin de limiter le nombre de licences il semble intéressant de n'utiliser qu'un seul utilisateur pour chaque interaction avec la GED.

**Utilisateurs Actes Office :**

- Droits utilisateurs pouvant envoyer en GED : droits de création d'un document final
- Droits utilisateurs pour visualiser les documents GED : agent
- Droits utilisateurs pour paramétrer la GED : administrateur.

### VII.2.2.4 CAS D'UTILISATION AVEC AOLINK

Dans le cas où le contrôle de légalité s'effectue depuis AoLink, il est intéressant de mettre en place un enregistrement du document final ainsi que du

bordereau d'acquittement. Pour les moments ce sont des éléments qui ne sont pas pris en compte par AoLink.

Il est donc intéressant de mettre en place l'enregistrement de ces éléments dans Actes Office depuis AoLink.

Il n'existe pour le moment aucun moyen de récupérer le document validé ou le bordereau depuis AoLink. Une fois ces documents enregistrés par AoLink il semble intéressant de les rendre visualisable par l'utilisateur.

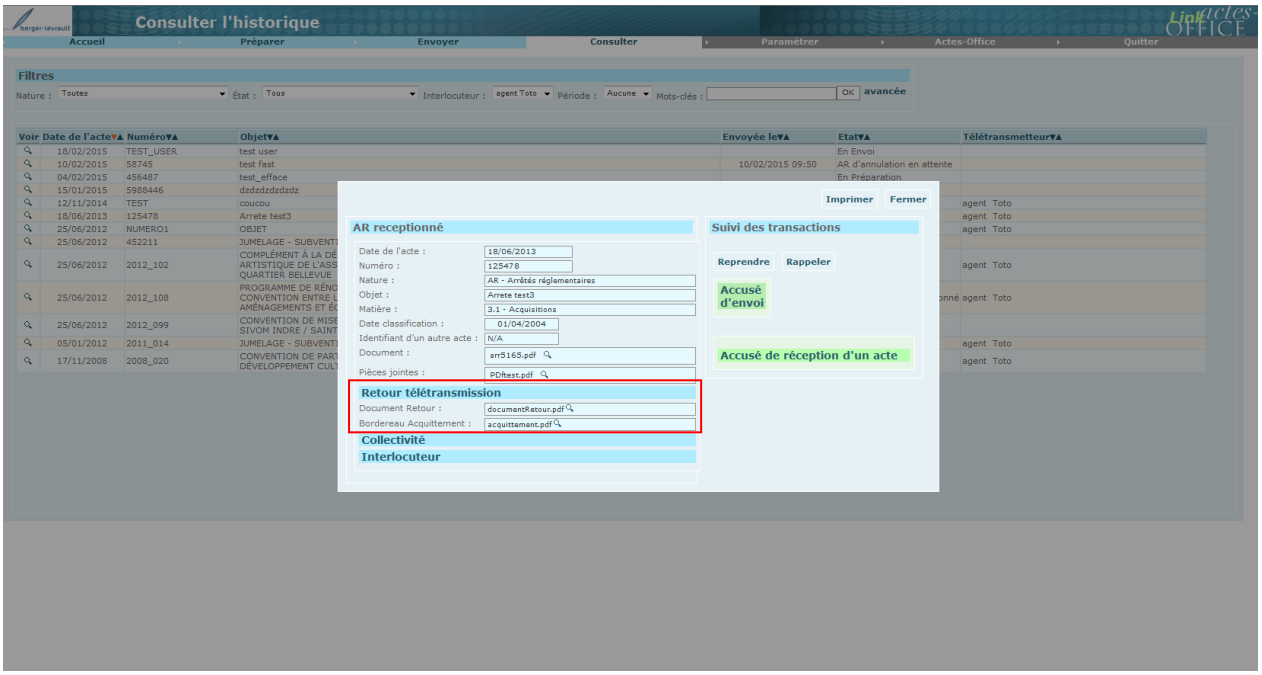


Figure 41 - Document Final Aolink

La visualisation de ces nouveaux documents est possible au niveau de l'onglet « consulter ». Lorsqu'on clique sur un acte la dialogue box ci-dessus s'affiche. En ajoutant une zone « retour télétransmission » avec une ligne « Document retour » et une autre « Bordereau d'acquittement » on permet la visualisation du document. Pour afficher le document il faut cliquer sur la loupe.

## VII.2 TESTS

Au même titre que pour une application, il est nécessaire de mettre en place des tests pour garantir la qualité, la sécurité, la non régression, la résistance à la charge.

Plusieurs différentes étapes de test jusqu'à la mise en production :

- les tests unitaires,
- les tests fonctionnels,
- les tests métier,
- les tests de charge/performance,
- mise en production.

Nous pouvons lancer les tests sur le server embedded d'Anypoint Studio mais il faut également les exécuter sur une instance standalone sur le poste de développement et/ou une PIC.

### VII.2.1 TESTS UNITAIRES

Tests custom Java pour :

- les *transformers*,
- les *components*.

Nous utilisons JUnit avec surcharge de classes de base pour les tests qui vont valider le cycle de vie du *transformer*/*component*.

On peut effectuer des assertions par rapport à un jeu de données ou un comportement attendu.

Cela permet de garantir la non régression des classes Java des composants.

Exemple test unitaire d'un transformer

Classe transformer par exemple pour créer une commande dans SalesForce à partir d'un message au format donné.

```
extends AbstractTransformerTestCase
```

Une méthode getTransformer() pour récupérer le transformer,

Une méthode getTestData() pour setup : création du message qui va être testé.

Une méthode testTransform : méthode triple A - Arrange, Act, Assert

- get testData
- sollicitation du transformer et récupération du résultat
- assertions sur le résultat de la transformation

### VII.2.2 TESTS FONCTIONNELS

On utilise Junit avec surcharge de classes de base, avec méthode setup pour le chargement de la configuration.

On utilise également l'API Mule Java, notamment le client Mule.

Parfois, pour les tests fonctionnels, on peut créer une entrée composite avec une entrée VM qui ne sera pas utilisé en production mais uniquement à des fins de test (peut-être avec un nommage spécifique à définir) qui permet de faire des tests sur les propriétés explicites nécessaires pour le bon fonctionnement du flow (exemple : ContentType dans une requête HTTP).

Il est parfois nécessaire de mocker les points de sortie :

- En incluant le module mockito pour mocker les point de sortie qui sera atteint par paramétrage/configuration du point de sortie,
- on peut également intégrer des flows Mule pour mocker les points de sortie.

On peut effectuer des assertions par rapport à un jeu de données ou un comportement attendu.

Cela permet de garantir la non régression des flows, et mesurer l'impact de l'arrivée de nouveaux flows.

Exemple d'un test fonctionnel

Test d'une API REST qui renvoie une liste d'utilisateurs.

```
extends FonctionalTestCase
```

Exemple de l'entrée VM private pour les tests :

```
vm://unit.test.internal.path?connector=vmSyncConnector
```

Utilisation du MuleClient (à n'utiliser que dans les classes de test ! problème de memory leaks ou impact sur la JVM) :

```
MuleClient client = new MuleClient(muleContext);

MuleMessage response = client.send(…);
```

Assertions : utilisation d'une jsonFilter (= new IsJsonFilter();) pour tester que la réponse est de type JSON

```
assertTrue(jsonFilter.accept(response));
```

### VII.2.3 TESTS METIERS

On déploie ensuite les applications Mule sur un serveur de recette avec maven, Jenkins…

Les référents métier peuvent tester en réel les fonctionnalités de l'ESB.

La découverte de bugs en recette renvoie vers les tests unitaires et/ou fonctionnels.

### VII.2.4 TESTS DE CHARGE/PERFORMANCE

Les tests de charge permettent :

- de valider la configuration Mule (pools de thread, timeout),
- de détecter les goulots d'étranglement, les surcharges CPU, les fuites mémoire,
- de déterminer les seuils de tolérance pour les réponses afin de correctement paramétrer le tuning Mule.

Ces tests sont importants à effectuer pour éviter les blocages en production :

- *short run* : 30/60 minutes,
- *long run* : 24/48 heures,
- avec montée en charge par *ramping* (en escalier) pour tester les temps de réponse (normalement, augmentation linéaire avec constance sur chaque pallier),
- avec charge constante : pour vérifier la non dégradation des temps de réponse,
- avec montée linéaire sans limite pour trouver le point de rupture de l'application.

En cas de problème détecté, il faut effectuer du tuning et relancer les tests de charge.

On peut effectuer les tests de charge/performance sur les applications déployées sur la PIC avec Gatling.

Dashboard avec déploiement de scenarios de tests sur des machines linux (injecteurs) qui vont stresser le serveur Mule, puis récupération des indicateurs jMeter pour le reporting.

Utilisation de :

- JStat pour les statistiques de la JVM,

- Monitoring mémoire, CPU.

**Mise en production**

Avec retour vers les tests de charge en cas de dysfonctionnement constaté en production (timeout, *deny of service*…) ou vers les tests unitaires/fonctionnels en cas de bug.

# V BILAN

## V.1 BILAN PERSONNEL

Sur un plan personnel, cette année de travail au sein de l'entreprise Berger-Levrault m'a permis de développer des compétences de gestion de projet, d'autonomie et d'expertise sur le logiciel Actes office.

La gestion de ce projet, a été particulière par le fait que l'équipe projet était principalement implantée à Nantes. La communication a donc été un challenge étant la seule personne sur ce projet à Toulouse. Mais également par l'architecture multi service du projet d'intégration au bus applicatif. En effet, ce projet est multi service, transverse et développé par des équipes dans toute la France.

Lorsque je suis arrivée sur le projet d'intégration d'Actes office à l'architecture SOA, le projet du bus applicatif était en phase de production, j'ai donc pu apprendre énormément sur la mise en place d'une telle architecture, et j'ai pu participer à ses phases de développement, tests et mise en production

Tout en étant impliquée sur le projet du bus applicatif on m'a permis de prendre une place primordiale dans l'équipe Acte office sur le module AoLink, en étant la seule personne de l'équipe en position d'expertise. Ce module ayant subi une perte de compétences au sein de l'entreprise, mon travail était de mettre en place des processus afin que cette situation ne se reproduise plus.

Cette expérience m'a aussi permis d'appréhender le monde de l'édition logicielle, mais aussi celui spécifique des collectivités. Notamment au niveau de l'adaptation aux réglementations particulières à la dématérialisation des actes.

Au niveau technique, la mise en place d'une architecture SOA a également été très enrichissante. Tout comme le développement de services dans l'ESB, qui m'ont permis d'acquérir des connaissances supplémentaires.

## V.2 BILAN GENERAL

Les apports pour l'entreprise de la mise en place de cette architecture sont principalement dans la capitalisation du temps de développement. Actes office ayant fait office de logiciel test pour cette architecture, nous pouvons en conclure que c'est une solution intéressante pour les problématiques d'interopérabilité.

Désormais lorsqu'un nouveau tiers de télétransmission sera demandé par une collectivité, un seul développement sera nécessaire pour l'ensemble de l'entreprise. Ce développement étant pris en charge par la cellule BLES, transverse, il ne sera pas imputé au logiciel qui en a besoin, et sera bénéficiale aux autres applications.

De plus, l'intégration de nouveaux services étant illimité, ce projet est bénéficiable à tous les logiciels de Berger-Levrault. En effet, en intégrant les services de dématérialisation et de GED nous avons pu démontrer que des services d'utilisation et de types différents peuvent être implémentés.

Cette logique permet également d'avoir une logique de services moins centrée sur le logiciel en lui-même. Le déport de la gestion des services à un module indépendant a pour but de décentrer le problème et de le rendre dépendant d'un service de l'entreprise plutôt que d'une équipe logiciel. Cette pratique, tout en consultant les besoins métiers, permet une vision plus globale sur le service et sur les sous services à mettre en place.

## V.3 PERSPECTIVES

Il existe cependant quelques limitations à ce système, notamment au niveau de la capitalisation du temps de développement. En effet, il semble intéressant de diminuer encore le temps de développement en mettant en place un module dans l'architecture permettant une génération automatique des services en fonction d'un paramétrage.

Cette solution a été envisagée cependant elle n'est possible que par l'utilisation des informations du service, que celui-ci communique, avec le wsdl pour

SOAP par exemple. Cependant, actuellement les services comme la GED ne sont pas développés en interne par Berger-Levrault mais des équipes indépendantes, ce qui pose un problème dans l'uniformisation ceux-ci. Par exemple BLGed n'expose pas ces services ce qui rend impossible la génération des services automatique.

Afin de résoudre cette problématique il serait alors nécessaire de repenser les collaborations et les cahiers des charges des logiciels pouvant devenir des services en amont.

# VI ANNEXES

## CLASSIFICATION DES ACTES

Les actes sont classés sous forme d'arborescence possédant cinq niveaux. Les deux premiers niveaux sont nationaux et sont obligatoires contrairement aux niveaux suivants qui seront paramétrés par chaque site exerçant le contrôle de légalité. Leur caractère obligatoire sera déterminé suivant l'accord entre la collectivité et le représentant de l'état.

**1. COMMANDE PUBLIQUE**

1.1. Marchés publics

1.2. Délégations de service public

*1.2.1. par type de contrat :*

*1.2.1.1. concession*

*1.2.1.2. affermage*

*1.2.1.3. autres contrats*

*1.2.2. Par catégorie de service public concerné*

*1.2.2.1. Service public industriel et commercial*

*1.2.2.1.1. eau, assainissement*

*1.2.2.1.2. élimination des déchets*

*1.2.2.1.3. pompes funèbres*

*1.2.2.2. Service public administratif*

*1.2.2.2.1. enseignement public*

*1.2.2.2.2. enseignement privé*

*1.2.2.2.3. action sociale*

1.3. Conventions de mandat

1.4. Autres contrats

1.5. Transactions (protocole d'accord transactionnel)

1.6. Maîtrise d'œuvre

1.7. Actes spéciaux et divers

**2. URBANISME**

2.1. Documents d'urbanisme

2.2. Actes relatifs au droit d'occupation ou d'utilisation des sols

2.3. Droit de préemption urbain

**3. DOMAINE et PATRIMOINE**

3.1. Acquisitions

3.2. Aliénations

3.3. Locations

3.4. Limites territoriales

3.5. Actes de gestion du domaine public

3.6. Autres actes de gestion du domaine privé

**4. FONCTION PUBLIQUE**

4.1. Personnels titulaires et stagiaires de la F.P.T.

4.2. Personnels contractuels

4.3. Fonction publique hospitalière

4.4. Autres catégories de personnels

4.5. Régime indemnitaire

**5. INSTITUTIONS et VIE POLITIQUE**

5.1. Election exécutif

5.2. Fonctionnement des assemblées

5.3. Désignation de représentants

5.4. Délégation de fonctions

5.5. Délégations de signature

5.6. Exercice des mandats locaux

5.7. Intercommunalité

5.8. Décision d'ester en justice

**6. LIBERTES PUBLIQUES et POUVOIRS DE POLICE**

6.1. Police municipale

6.2. Pouvoirs du président du conseil général

6.3. Pouvoirs du président du conseil régional

6.4. Autres actes réglementaires

6.5. Actes pris au nom de l'Etat

**7. FINANCES LOCALES**

7.1. Décisions budgétaires (B.P., D.M., C.A.…)

7.2. Fiscalité

7.3. Emprunts

7.4. Interventions économiques

7.5. Subventions

7.6. Contributions budgétaires

7.7. Avances

7.8. Fonds de concours

7.9. Prise de participation (SEM,etc.)

7.10. Divers

**8. DOMAINES DE COMPETENCES PAR THEMES**

8.1. Enseignement

8.2. Aide sociale

8.3. Voirie

8.4. Aménagement du territoire

8.5. Politique de la ville, habitat, logement

8.6. Emploi, formation professionnelle

8.7. Transports

8.8. Environnement

8.9. Culture

**9. AUTRES DOMAINES DE COMPETENCES**

9.1. Autres domaines de compétence des communes

9.2. Autres domaines de compétence des départements

9.3. Autres domaines de compétence des régions

9.4. Voeux et motions

# DESCRIPTION DES SERVICES GED

### Service Echo

Appelle un des web services de la GED afin de vérifier que le paramétrage est correct, et que le serveur est UP. Le code retour HTTP est 200 si tout est OK.

Tableau 5 - service echo

| En entrée | |
|---|---|
| **URL** | /ged/echo |
| **Méthode HTTP** | GET |
| **Paramètres obligatoires** | clientId, Authorization |
| **En retour** | |
| **Format** | JSON |

### Service Get URL

Génère une URL de téléchargement pour un document en GED.

Tableau 6 - service get url

| En entrée | |
|---|---|
| **URL** | /ged/geturl |
| **Méthode HTTP** | GET |
| **Paramètres obligatoires** | clientId, Authorization, serviceData |
| **serviceData : paramètres facultatifs** | "username":"[utilisateur]","password":"[mot de passe]" |
| **En retour** | |
| **Format** | JSON |
| **Contenu du champ content** | URL permettant de télécharger le document demandé. |

Service Path

Permet d'interagir avec un document ou un répertoire sur la GED en fonction de son chemin.

En fonction de la méthode HTTP utilisée, on peut respectivement récupérer les métadonnées, créer, mettre à jour ou supprimer un document ou répertoire à l'aide de son chemin.

Méthode GET

Tableau 7 - service path GET

| En entrée | |
|---|---|
| **URL** | /ged/path/[chemin vers le document/répertoire] |
| **Description** | Récupération des métadonnées |
| **Paramètres obligatoires** | clientId, Authorization, serviceData |
| **serviceData : paramètres obligatoires** | "objecttype":"DOCUMENT" ou "objecttype":"FOLDER" |
| **serviceData : paramètres facultatifs** | "username":"[utilisateur]" "password":"[mot de passe]" |
| **En retour** | |
| **Format** | JSON |
| **Contenu du champ content** | Métadonnées du document ou du répertoire au format clé/valeur en JSON |

Méthode POST

Tableau 8 - service path post

| En entrée | |
|---|---|
| **URL** | /ged/path/[chemin vers le répertoire parent] |
| **Description** | Récupération des métadonnées |
| **Paramètres obligatoires** | clientId, Authorization, serviceData |
| **serviceData : paramètres obligatoires** | "objecttype":"DOCUMENT" ou "objecttype":"FOLDER" "objectname":"[nom du répertoire à créer]", si |

| | |
|---|---|
| | objecttype est positionné à FOLDER, ce paramètre est obligatoire. Sinon il sera ignoré. |
| **serviceData : paramètres facultatifs** | "username":"[utilisateur]" "password":"[mot de passe]" "metadata":"[métadonnées]", métadonnées à associer à l'objet qui va être créé, au format JSON : couples de clé/valeurs. |
| **multipart/form-data** | Document à déposer, son nom sera le nom du fichier passé en multipart/form-data. Ce paramètre sera ignoré dans le cas d'une création de répertoire. |
| **En retour** | |
| **Format** | JSON |
| **Contenu du champ content** | Identifiant de l'objet créé sous forme de chaîne de caractère. |

Méthode PUT

Tableau 9 - service path put

| **En entrée** | |
|---|---|
| **URL** | /ged/path/[chemin vers le document ou le répertoire] |
| **Méthode HTTP** | PUT |
| **Description** | Mise à jour d'un document ou d'un répertoire Attention, la mise à jour d'un répertoire n'est pas supportée sur les GEDs suivantes : Alfresco, Sharepoint et Nuxeo |
| **Paramètres obligatoires** | clientId, Authorization, serviceData |
| **serviceData : paramètres obligatoires** | "objecttype":"DOCUMENT" ou "objecttype":"FOLDER" "objectname":"[nouveau nom du répertoire]", si objecttype est positionné à FOLDER, ce |

| | |
|---|---|
| | paramètre  est obligatoire. Sinon il sera ignoré. |
| **serviceData : paramètres facultatifs** | "username":"[utilisateur]"<br>"password":"[mot de passe]"<br>"metadata":"[métadonnées]", métadonnées à associer à l'objet qui va être mis à jour, au format JSON : couples de clé/valeurs. |
| **multipart/form-data** | Document à déposer, son nom sera le nom du fichier passé en multipart/form-data. Ce paramètre sera ignoré dans le cas d'une mise à jour de répertoire. |
| **En retour** | |
| **Format** | JSON |
| **Contenu du champ content** | Identifiant de l'objet sous forme de chaîne de caractère. |

Méthode DELETE

Tableau 10 - service delete

| **En entrée** | |
|---|---|
| **URL** | /ged/path/[chemin vers le document/répertoire] |
| **Description** | Suppression d'un document ou un répertoire |
| **Paramètres obligatoires** | clientId, Authorization, serviceData |
| **serviceData : paramètres obligatoires** | "objecttype":"DOCUMENT" ou<br>"objecttype":"FOLDER" |
| **serviceData : paramètres facultatifs** | "username":"[utilisateur]"<br>"password":"[mot de passe]" |
| **En retour** | |
| **Format** | JSON |
| **Contenu du champ content** | Métadonnées du document ou du répertoire au format clé/valeur en JSON |

Service Id

Même service que Path, mais avec l'identifiant d'un document ou d'un répertoire (identifiant du répertoire parent pour les créations).

Méthode GET

Tableau 11 - service Id get

| En entrée | |
|---|---|
| **URL** | /ged/id/[identifiant GED du document/répertoire] |
| **Description** | Récupération des métadonnées |
| **Paramètres obligatoires** | clientId, Authorization, serviceData |
| **serviceData : paramètres obligatoires** | "objecttype":"DOCUMENT" ou "objecttype":"FOLDER" |
| **serviceData : paramètres facultatifs** | "username":"[utilisateur]" "password":"[mot de passe]" |
| **En retour** | |
| **Format** | JSON |
| **Contenu du champ content** | Métadonnées du document ou du répertoire au format clé/valeur en JSON |

Méthode POST

Tableau 12 - service id post

| En entrée | |
|---|---|
| **URL** | /ged/id/[identifiant du répertoire parent] |
| **Description** | Récupération des métadonnées |
| **Paramètres obligatoires** | clientId, Authorization, serviceData |
| **serviceData : paramètres obligatoires** | "objecttype":"DOCUMENT" ou "objecttype":"FOLDER" "objectname":"[nom du répertoire à créer]", si objecttype est positionné à FOLDER, ce paramètre est obligatoire. Sinon il sera ignoré. |

| | |
|---|---|
| **serviceData : paramètres facultatifs** | "username":"[utilisateur]" "password":"[mot de passe]" "metadata":"[métadonnées]", métadonnées à associer à l'objet qui va être créé, au format JSON : couples de clé/valeurs. |
| **multipart/form-data** | Document à déposer, son nom sera le nom du fichier passé en multipart/form-data. Ce paramètre sera ignoré dans le cas d'une création de répertoire. |
| | En retour |
| **Format** | JSON |
| **Contenu du champ content** | Identifiant de l'objet créé sous forme de chaîne de caractère. |

Méthode PUT

| | |
|---|---|
| | En entrée |
| **URL** | /ged/id/[identifiant du document/répertoire] |
| **Méthode HTTP** | PUT |
| **Description** | Mise à jour d'un document ou d'un répertoire Attention, la mise à jour d'un répertoire n'est pas supporté sur les GEDs suivantes : Alfresco, Sharepoint et Nuxeo |
| **Paramètres obligatoires** | clientId, Authorization, serviceData |
| **serviceData : paramètres obligatoires** | "objecttype":"DOCUMENT" ou "objecttype":"FOLDER" "objectname":"[nouveau nom du répertoire]", si objecttype est positionné à FOLDER, ce paramètre est obligatoire. Sinon il sera ignoré. |
| **serviceData : paramètres facultatifs** | "username":"[utilisateur]" "password":"[mot de passe]" "metadata":"[métadonnées]", métadonnées à |

| | |
|---|---|
| | associer à l'objet qui va être mis à jour, au format JSON : couples de clé/valeurs. |
| **multipart/form-data** | Document à déposer, son nom sera le nom du fichier passé en multipart/form-data. Ce paramètre sera ignoré dans le cas d'une mise à jour de répertoire. |
| | En retour |
| **Format** | JSON |
| **Contenu du champ content** | Identifiant de l'objet sous forme de chaîne de caractère. |

Méthode DELETE

| | |
|---|---|
| | En entrée |
| **URL** | /ged/id/[identifiant du document/répertoire] |
| **Description** | Suppression d'un document ou un répertoire |
| **Paramètres obligatoires** | clientId, Authorization, serviceData |
| **serviceData : paramètres obligatoires** | "objecttype":"DOCUMENT" ou "objecttype":"FOLDER" |
| **serviceData : paramètres facultatifs** | "username":"[utilisateur]" "password":"[mot de passe]" |
| | En retour |
| **Format** | JSON |
| **Contenu du champ content** | Métadonnées du document ou du répertoire au format clé/valeur en JSON |

Service Content by Path

Récupération du contenu d'un document ou d'un répertoire par chemin.

Méthode GET

<table>
<tr><th colspan="2">En entrée</th></tr>
<tr><td>URL</td><td>/ged/contentbypath/[chemin du document/répertoire]</td></tr>
<tr><td>Description</td><td>Récupération du contenu, format flux de données pour un document, format JSON pour un répertoire</td></tr>
<tr><td>Paramètres obligatoires</td><td>clientId, Authorization, serviceData</td></tr>
<tr><td>serviceData : paramètres obligatoires</td><td>"objecttype":"DOCUMENT" ou "objecttype":"FOLDER"</td></tr>
<tr><td>serviceData : paramètres facultatifs</td><td>"username":"[utilisateur]"<br>"password":"[mot de passe]"</td></tr>
<tr><th colspan="2">En retour</th></tr>
<tr><td>Format</td><td>JSON si répertoire, flux direct si document</td></tr>
<tr><td>Contenu du champ content dans le cas d’un répertoire</td><td>JSON, sous forme de tableau d’objets, chaque objet contient les propriétés suivantes :
<table>
<tr><th>cmis:name</th><th>Nom de l’objet</th></tr>
<tr><td>cmis:objectId</td><td>Identifiant GED de l’objet</td></tr>
<tr><td>cmis:objectTypeId</td><td>Type de l’objet, DOCUMENT ou FOLDER</td></tr>
<tr><td>nbchildren</td><td>Nombre d’enfants de l’objet, contient 0 dans le cas d’un document</td></tr>
<tr><td>cmis:contentStreamLength</td><td>Taille de l’objet en octets, contient 0 dans le cas d’un répertoire. Attention, contiendra 0 si la GED est Openbee</td></tr>
</table>
</td></tr>
</table>

Service Content by Id

Même service que Content By Path mais à l'aide d'un identifiant.

Méthode GET

<table>
<tr><th colspan="2">En entrée</th></tr>
<tr><td>URL</td><td>/ged/contentbyid/[identifiant du document/répertoire]</td></tr>
<tr><td>Description</td><td>Récupération du contenu, format flux de données pour un document, format JSON pour un répertoire</td></tr>
<tr><td>Paramètres obligatoires</td><td>clientId, Authorization, serviceData</td></tr>
<tr><td>serviceData : paramètres obligatoires</td><td>"objecttype":"DOCUMENT" ou "objecttype":"FOLDER"</td></tr>
<tr><td>serviceData : paramètres facultatifs</td><td>"username":"[utilisateur]"<br>"password":"[mot de passe]"</td></tr>
<tr><th colspan="2">En retour</th></tr>
<tr><td>Format</td><td>JSON si répertoire, flux direct si document</td></tr>
<tr><td>Contenu du champ content dans le cas d’un répertoire</td><td>JSON, sous forme de tableau d’objets, chaque objet contient les propriétés suivantes :
<table>
<tr><th>cmis:name</th><th>Nom de l’objet</th></tr>
<tr><td>cmis:objectId</td><td>Identifiant GED de l’objet</td></tr>
<tr><td>cmis:objectTypeId</td><td>Type de l’objet, DOCUMENT ou FOLDER</td></tr>
<tr><td>nbchildren</td><td>Nombre d’enfants de l’objet, contient 0 dans le cas d’un document</td></tr>
<tr><td>cmis:contentStreamLength</td><td>Taille de l’objet en octets, contient 0 dans le cas d’un répertoire. Attention, contiendra 0 si la GED est Openbee</td></tr>
</table>
</td></tr>
</table>

Service Query request

Requête dans les documents de la GED.

Méthode GET

| En entrée | |
|---|---|
| **URL** | /ged/query |
| **Description** | Permet de requêter sur les documents ou répertoires de la GED |
| **Paramètres obligatoires** | clientId, Authorization, serviceData |
| **serviceData : paramètres obligatoires** | "statement", contient la requête à exécuter, voir les specs CMIS pour plus de détails.<br>Exemple :<br>SELECT cmis:name, cmis:objectId FROM cmis:document WHERE cmis:name = 'flux.pdf' |
| **serviceData : paramètres facultatifs** | "username":"[utilisateur]"<br>"password":"[mot de passe]" |
| **En retour** | |
| **Format** | JSON |
| **Contenu du champ content** | JSON, sous forme de tableau d'objets, chaque objet contient les propriétés sélectionnées par la requête |

Service Query contains request

Recherche de contenu dans les métadonnées et le contenu des documents de la GED.

Méthode GET

| En entrée | |
|---|---|
| **URL** | /ged/querycontains |
| **Description** | Permet de requêter sur les documents de la GED à l'aide d'un champ unique |
| **Paramètres obligatoires** | clientId, Authorization, serviceData |
| **serviceData : paramètres obligatoires** | "select", contient sous forme de liste JSON les champs à sélectionner<br>"properties", contient une liste JSON des champs sur lesquels on veut chercher<br>"value", valeur à rechercher dans les champs précisés par le paramètre précédent |
| **serviceData : paramètres facultatifs** | "username":"[utilisateur]"<br>"password":"[mot de passe]" |
| **En retour** | |
| **Format** | JSON |
| **Contenu du champ content** | JSON, sous forme de tableau d'objets, chaque objet contient les propriétés sélectionnées par la requête (voir paramètre "select") |

# VII BIBLIOGRAPHIE

VERNADAT, François B. Interoperable enterprise systems: Principles, concepts, and methods. *Annual Reviews in Control*, 2007, vol. 31, no 1, p. 137-145.

BREBNER, Paul. Service-oriented performance modeling the mule enterprise service bus (esb) loan broker application. In : *Software Engineering and Advanced Applications, 2009. SEAA'09. 35th Euromicro Conference on*. IEEE, 2009. p. 404-411.

RIETSCH, Jean-Marc, CHABIN, Marie-Anne, et CAPRIOLI, Éric. Dématérialisation et archivage électronique. *Paris: Dunod*, 2006.

Collectivite-locales. ACTES - Dématérialisation de la transmission des actes. [en ligne]. http://www.collectivites-locales.gouv.fr/actes-0 [14/10/2015].

# VIII LISTE DES FIGURES

# IX LISTE DES TABLEAUX

Extension du logiciel actes office par intégration dans une architecture SOA.

Mémoire d'Ingénieur C.N.A.M., Toulouse 2015.

---


**RESUME**

L'architecture SOA est de plus en plus utilisée dans les entreprises. Cette Architecture Orientée Service, dont le terme est apparu au cours de la période 2000-2001, est depuis un modèle au sein des systèmes d'informations et dans l'informatique en général. La mise en place de ce genre d'architecture permet de facilité l'extension logicielle, sujet récurrent dans le monde du génie logiciel. En effet, toute application a pour vocation d'être étendue pour des besoins fonctionnels.

Une des solutions à envisager est l'intégration de ce logiciel dans une architecture lui permettant d'isoler ses besoins de services. Le service devient donc externe et factorisable à d'autres besoins. Pour un éditeur de logiciel cette factorisation est un gain de temps énorme, puisque le service n'est plus dépendant de l'applicatif. C'est la démarche que Berger-Levrault a souhaité mettre en œuvre pour l'ensemble de ses logiciels.



Mots clés : SOA, services, extension logicielle, interopérabilité, ESB.


---


**SUMMARY**

SOA is increasingly used in businesses. This service-oriented architecture, term originated during the period 2000-2001, has been a model in the information and computer systems in general. The implementation of this type of architecture allows easy software extension, recurring topic in the world of software engineering. In fact, any application aims to be extended to functional needs.

One solution to consider is the integration of this software in an architecture allowing it to isolate its service needs. The service is thus external and usable for other needs. For a software company that factoring is a huge time saver, since the service is no longer dependent on the application. This is the approach that Berger-Levrault wished to implement for all of its software.



Key words: SOA, services, software extensibility, interoperability, ESB.