

h e g

Haute école de gestion
Genève

Mule ESB



Travail de Bachelor réalisé en vue de l'obtention du Bachelor HES

par :

Djavan SERGENT

Conseiller au travail de Bachelor :

Peter Daehne, Professeur HES

Genève, 11.05.2015

Haute École de Gestion de Genève (HEG-GE)

Filière Informatique de gestion

Déclaration

Ce travail de Bachelor est réalisé dans le cadre de l'examen final de la Haute école de gestion de Genève, en vue de l'obtention du titre de Bachelor en Informatique de Gestion.

L'étudiant a envoyé ce document par email à l'adresse remise par son conseiller au travail de Bachelor pour analyse par le logiciel de détection de plagiat URKUND, selon la procédure détaillée à l'URL suivante : http://www.orkund.fr/student_gorsahar.asp.

L'étudiant accepte, le cas échéant, la clause de confidentialité. L'utilisation des conclusions et recommandations formulées dans le travail de Bachelor, sans préjuger de leur valeur, n'engage ni la responsabilité de l'auteur, ni celle du conseiller au travail de Bachelor, du juré et de la HEG.

« J'atteste avoir réalisé seul le présent travail, sans avoir utilisé des sources autres que celles citées dans la bibliographie. »

Fait à Genève, le 11.05.2015

Djavan SERGENT

Remerciements

Je tiens à remercier toutes les personnes qui ont rendu ce projet possible, en commençant par M. Peter Daehne, directeur de ce travail, M. Stéphane Labaye qui a supervisé le stage ainsi que M. François Bonvallat, responsable du projet et du développement.

Je remercie également les collaborateurs de la RTS pour leurs explications et leurs conseils avisés.

Enfin, je remercie tous ceux qui m'ont relu, corrigé, conseillé et soutenu.

Résumé

Ce document a pour but de faire un tour d'horizon des fonctionnalités de Mule ESB. Il est structuré comme ceci :

1. Une petite introduction sur la notion d'ESB.
2. Une étude de marché et une analyse des besoins.
3. Une présentation du bus choisi.
4. Une explication sur la méthode de modélisation des prototypes.
5. Une présentation des prototypes réalisés ainsi que leur analyse.
6. Une explication de la mise en production.
7. Une évaluation des prototypes.
8. Une conclusion technique.
9. Une évaluation personnelle.
10. Un glossaire.

Au travers de ces différents chapitres, *Mule ESB 3.6.1* est testé afin d'en déterminer les points forts/faibles et les possibilités de migration pour les processus déjà en place au sein de la RTS.

Bonne lecture !

Table des matières

Déclaration	i
Remerciements	ii
Résumé	iii
Liste des tableaux	vi
Liste des figures	vi
1. Introduction	1
2. Analyse de marché	2
2.1 Définition des besoins	2
2.2 Plan d'analyse	3
2.3 Analyse des résultats et recommandations	3
3. Présentation du bus : Mule ESB	5
3.1 Prise en main	5
3.2 Messages	6
3.3 Éléments du flux	7
3.3.1 Connecteurs.....	7
3.3.2 Transformateurs.....	8
3.3.3 Composants.....	8
3.3.4 Étendues.....	9
3.3.5 Filtres.....	9
3.3.6 Contrôles de flux	10
3.3.7 Gestion des erreurs.....	10
3.4 Les nouveautés	10
3.4.1 Anypoint Studio	11
3.4.2 DataSense Language Query	12
3.4.3 Project Environment Settings	13
3.4.4 CloudHub Sandbox.....	13
3.4.5 Watermarks.....	13
3.4.6 Prochaine version	14
3.5 Langages et développement	14
3.5.1 XML – XSLT.....	14
3.5.2 Java	14
3.5.3 MEL	15
3.5.4 Expressions régulières.....	15
3.5.5 Fichiers « .properties ».....	15
3.5.6 Autres possibilités	15
4. Modélisation des flux	16
4.1 Principe de modélisation standardisée	16

4.2	Outils et logiciels	16
4.2.1	ArgoUML.....	16
4.2.2	Draw IO.....	17
4.3	Exemples de modèles.....	18
5.	Prototypes.....	19
5.1	Éléments génériques	20
5.2	N°1 : Rtsinfochip	21
5.3	N°2 : Thumbfile Dispatch.....	25
5.4	N°3 : Betasuisse (JSON).....	28
5.5	N°4 : Betasuisse (XML)	33
5.6	N°5 : Move and Rename.....	37
5.7	N°6 : Import Récuratif	41
5.8	N°7 : Thumbfile Requester	45
5.9	N°8 : Sparkex.....	49
5.10	N°9 : MassDL.....	52
6.	Mise en production.....	60
6.1	Fichiers de propriétés.....	60
6.2	Gestion d'erreurs	60
6.3	Transformateurs globaux	61
6.4	Runtime CE.....	62
6.5	Monitoring	62
6.5.1	MMC	62
6.5.2	JMX.....	63
6.6	Le flux console.....	64
7.	Evaluation des prototypes.....	68
7.1	Tests	68
7.2	Récapitulatif des composants utilisés	74
7.3	Problèmes rencontrés	78
8.	Conclusion	79
9.	Evaluation personnelle	81
10.	Glossaire	82
	Bibliographie	83
	Annexe 1 : Tableau comparatif des ESB.....	86
	Annexe 2 : Console Management JMX.....	87
	Annexe 3 : Console Management JMX.....	88

Liste des tableaux

Tableau 1 : Critères et besoins.....	2
Tableau 2 : Représentation des flux.....	18
Tableau 3 : Tableaux récapitulatifs des tests.....	68
Tableau 4 : Récapitulatif des composants utilisés	75

Liste des figures

Figure 1 : Logos des ESB.....	4
Figure 2 : Tutoriels officiels Mule ESB.....	5
Figure 3 : Composition d'un Message Mule.....	6
Figure 4 : Transport de propriétés Inbound	7
Figure 5 : Transport de propriétés Outbound.....	7
Figure 6 : Schéma général d'un flux.....	7
Figure 7 : Interface Anypoint Studio	11
Figure 8 : Documentation automatique générée par Mule	12
Figure 9 : Interface Query Builder.....	12
Figure 10 : Gestion d'environnements multiples	13
Figure 11 : Interface ArgoUML	17
Figure 12 : Interface Draw IO	17
Figure 13 : Exemple de modélisation UML	18
Figure 14 : Exemple de vue « Message Flow »	18
Figure 15 : Exemple de vue « Configuration XML ».....	18
Figure 16 : Modèle UML Rtsinfochip	22
Figure 17 : Modèle Mule Rtsinfochip	22
Figure 18 : Modèle UML Thumbfile Dispatch.....	26
Figure 19 : Modèle Mule Thumbfile Dispatch.....	26
Figure 20 : Modèle UML Betasuisse JSON	29
Figure 21 : Modèle Mule Betasuisse JSON	29
Figure 22 : Modèle UML Betasuisse XML	34
Figure 23 : Modèle Mule Betasuisse XML	34
Figure 24 : Modèle UML Move and Rename	38
Figure 25 : Modèle Mule Move and Rename.....	38
Figure 26 : Modèle UML Import Récursif	42
Figure 27 : Modèle Mule Import Récursif.....	42
Figure 28 : Modèle UML Thumbfile Requester	46
Figure 29 : Modèle Mule Thumbfile Requester.....	47
Figure 30 : Modèle UML Sparkex	50
Figure 31 : Modèle Mule Sparkex.....	50
Figure 32 : Modèle UML MassDL	54
Figure 33 : Modèle Mule MassDL.....	55
Figure 34 : Gestion d'erreurs et définition d'un transformateur global.....	61
Figure 35 : Changement de runtime	62
Figure 36 : Mule Management Console.....	63
Figure 37 : Modèle UML Management-Console	65
Figure 38 : Modèle Mule Management-Console	65

1. Introduction

« L'ESB est une nouvelle architecture qui exploite les services web, les systèmes orientés messages, le routage intelligent et la transformation. L'ESB agit comme une colonne vertébrale légère et omniprésente de l'intégration à travers laquelle les services logiciels et les composants applicatifs circulent »

Roy W. Schulte, Gartner Inc, 2003.

Aujourd'hui, les éditeurs sont multiples et les applications très nombreuses. Toutes les possibilités offertes par ces solutions n'ont pour limite que leur communication entre elles. En effet, n'étant pas éditées par les mêmes acteurs, on se retrouve très rapidement face à des applications qui ne peuvent pas échanger d'informations.

Le rôle du bus de service d'entreprise (ESB) est de combler ce gap. L'ESB représente une technique informatique intergicielle chargée d'assurer la communication entre des applications qui n'ont à l'origine pas été conçues pour fonctionner ensemble. Cette technique est basée sur plusieurs principes tels que les annuaires sémantiques partagés ou la communication par messages. Un ESB, c'est avant tout une architecture.

Les intérêts des ESB sont multiples, et comportent notamment la standardisation des concepts (XML, JMS, Web Service...), le routage intelligent, une architecture distribuée orientée service ou encore une grande fiabilité grâce à l'absence de point individuels de défaillance. Tous ces avantages permettent une excellente souplesse et une grande précision dans leur intégration au sein d'une entreprise.

Ce travail de Bachelor, effectué pour la RTS, met en pratique l'intégration de ces concepts dans la réalisation de prototypes simples qui permettent de migrer de la version actuellement en place de l'ESB à la dernière version en date, *Mule 3.6*. La plupart du temps, il s'agit de traiter de petits fichiers de métadonnées. Par exemple, faire communiquer la gestion logicielle des archives avec le robot physique contenant la très grande majorité des archives, en passant par le Web Service qui gère la base de données et les autorisations.

Les prototypes ont pour but de mettre en avant les points forts et faibles de cette nouvelle version et de définir le potentiel du bus d'entreprise Mule édité par Mulesoft, tout en le comparant aux autres acteurs majeur du domaine tel que WSO2 ou Talend dans une analyse de marché.

2. Analyse de marché

L'analyse de marché s'est portée sur les différentes solutions pouvant répondre aux besoins de la RTS. Les produits ont été évalués selon les critères énoncés ci-dessous sur la base des informations disponibles sur Internet (sites officiels, forums, évaluations par des tiers).

2.1 Définition des besoins

Tableau 1 : Critères et besoins

Critère	Besoin
Fonctionnalités, extensibilité et flexibilité	Le produit doit comporter un maximum de fonctionnalités courantes aux ESB (connecteurs, transformateurs, contrôleurs de flux etc.). Il doit également permettre d'implémenter des fonctionnalités qui ne lui seraient pas natives.
Facilité d'utilisation	Le produit doit être facile d'utilisation et permettre de créer/améliorer de nouveaux processus rapidement.
Communauté et documentation	Le produit doit être activement soutenu et doit bénéficier d'une documentation de qualité. C'est le critère principal.
Connecteurs disponibles	Minimum : WS – File – REST – HTTP – FTP – JDBC.
Prix	Les solutions propriétaires ne sont pas exclues pour autant que leurs fonctions justifient leur prix.
Performances	Il ne s'agit que de faire transiter de petits fichiers de métadonnées. La performance n'est donc pas un critère déterminant.
Fiabilité	La robustesse et la tolérance aux pannes du produit sont d'importants facteurs à prendre en compte.
Sécurité	Aucun besoin lié à la sécurité.

2.2 Plan d'analyse

Les données recueillies sont de type qualitatif. Comme il existe beaucoup de bus de services d'entreprise différents, tant open source que propriétaires, l'étude se porte principalement sur les plus connus. Six bus ont été sélectionnés sur la base de leur notoriété et de la quantité d'information disponible :

Nom	Editeur	Type
Mule ESB 3.5	MuleSoft	Open source
Talend	Talend	Open source
Fuse	Red Hat	Open source
WSO2	WSO2	Open source
Biztalk	Microsoft	Propriétaire
WebSphere	IBM	Propriétaire

La différence entre le nombre de bus open source et propriétaires s'explique par le fait que l'on privilégie une solution libre.

Les informations sont regroupées dans un tableau¹ afin de permettre de comparer les fonctionnalités de chaque ESB sur une base commune. La liste des fonctions n'est donc pas exhaustive et se concentre sur les fonctions importantes et non standard.

2.3 Analyse des résultats et recommandations

En analysant objectivement les résultats, on se rend compte de deux faits importants du point de vue de la collecte des informations :

Premièrement, la quantité d'informations disponible varie très largement d'un ESB à un autre. La plupart d'entre eux, surtout les logiciels propriétaires, sont peu précis quant à leurs fonctions et doivent être utilisés pour prendre pleinement conscience de leur potentiel. Malheureusement, il est impossible de tester chacun d'eux dans le cadre du projet et il faut donc se fier aux observations et aux tests d'autres personnes.

Deuxièmement, les informations provenant en grande partie des sites officiels, il est difficile d'obtenir un avis objectif sur le produit. Chaque éditeur propose un ensemble de fonctions, de connecteurs (voir 3.3.1 ci-dessous) et une interface graphique plus ou

¹ Voir annexe 1

moins facile à prendre en main qu'ils vantent comme étant la meilleure offre du marché.

Cependant, certains points forts et faibles sont tout de même révélés par cette analyse. Par exemple, la suite Talend aurait pu être un bon choix de par son écosystème et ses fonctions d'entreprise mais sa conception ETL (utilisée pour les gros traitements batch par exemple) orienté ESB ne correspond pas au besoin. Concernant les logiciels propriétaires, ils ne sont pas assez transparents dans leur description pour justifier ce prix de licence excessivement élevé. Il apparaît par exemple que la solution IBM coûterait au minimum 200'000.- par année, et cela dans sa configuration minimale.

Au final, Mule ESB se distingue de par la quantité d'informations disponibles, l'avis des internautes, la notoriété, l'expérience ainsi que sa facilité de prise en main. De plus, c'est l'ESB qui dispose, à ce jour, de la communauté la plus active et dont la documentation est la plus facilement exploitable. Enfin, c'est le logiciel en place dans sa version 2.5 à la RTS (version actuellement testée : 3.6.1).

Il convient tout de même de mettre en avant le faible nombre de fonctions d'entreprise (management, gestion des performances, règles métier etc.) et la difficulté à obtenir un prix pour une éventuelle utilisation de la version payante. De plus, son composant le plus important, le DataMapper qui permet de convertir un format de données en n'importe quel autre, fait partie de la version entreprise.

Figure 1 : Logos des ESB



3. Présentation du bus : Mule ESB

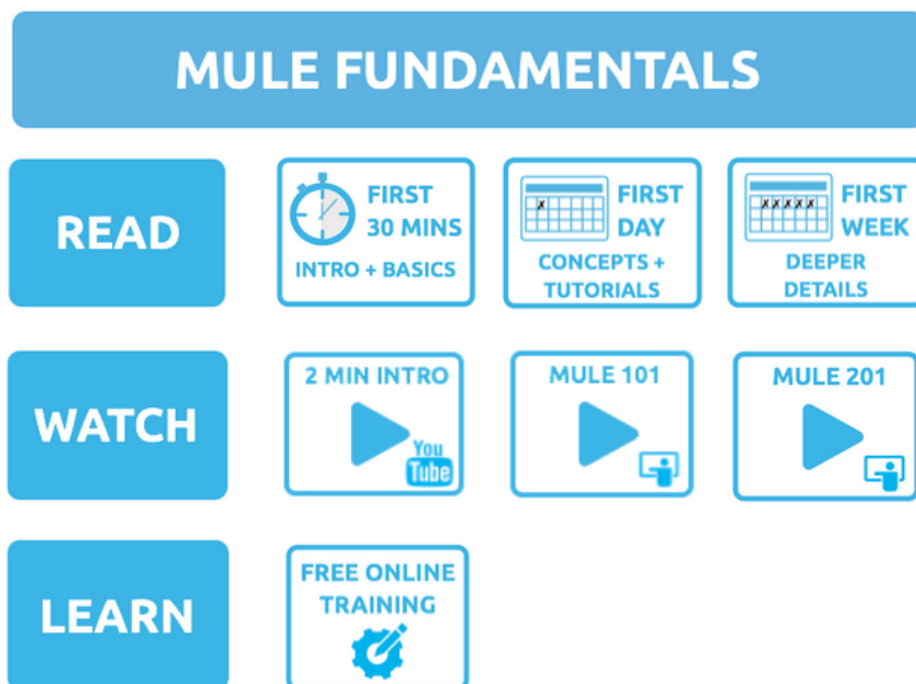
Mule ESB, développé par MuleSoft et actuellement dans sa version 3.6, est un bus de service d'entreprise qui bénéficie d'une large communauté et d'une excellente documentation. C'est ce produit qui a été sélectionné lors de l'analyse de marché pour ses nombreux avantages : communauté, facilité d'utilisation, maturité du produit, excellente documentation etc. Il est orienté dans la gestion des flux. Afin de bien comprendre les concepts de messages et de flux, il faut analyser le fonctionnement de Mule ESB.

3.1 Prise en main

Sur le site officiel de Mule, on peut trouver dans la documentation une série de tutoriels pour la prise en main du produit. Cela passe par 3 étapes : les 30 premières minutes avec Mule, la première journée avec Mule et la première semaine avec Mule. Au fil de ces tutoriels, on apprend à utiliser Anypoint Studio, à utiliser différents connecteurs (voir 3.3.1 ci-dessous), à faire des traitements simples ou encore à déployer son application.

Dans l'ensemble, ces guides sont de bonne qualité et offrent la possibilité aux novices de comprendre les concepts basiques de Mule ainsi que la façon de mettre en place le bus d'entreprise.

Figure 2 : Tutoriels officiels Mule ESB



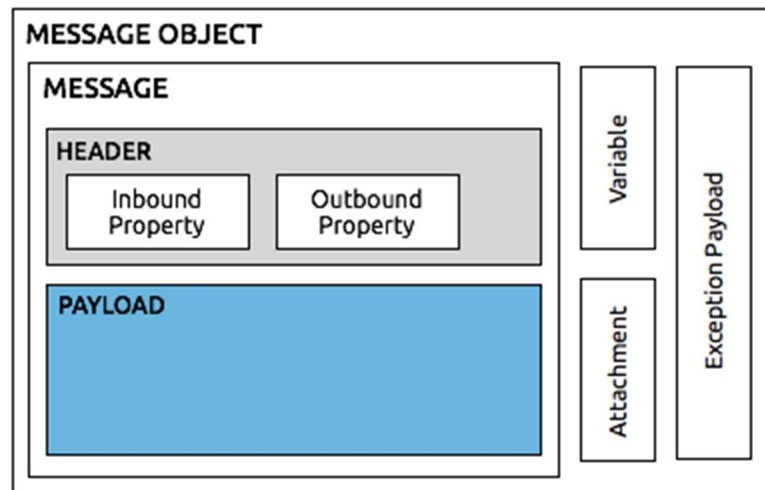
<http://www.mulesoft.org/documentation/display/current/Mule+Fundamentals>

3.2 Messages

Un Message Mule est une donnée qui va être transportée par un ou plusieurs flux au travers d'éléments qui forment une application. Ce message comporte deux parties principales :

- **Header** : contient les métadonnées du message.
- **Payload** : contient les données (exemple : un fichier)²

Figure 3 : Composition d'un Message Mule



<http://www.mulesoft.org/documentation/display/current/Mule+Message+Structure>

Les éléments *Attachement* et *Exception Payload* sont présents au sein d'un *Mule Message Object*, mais ceux-ci ne sont que très rarement manipulés.

Le *header* comporte deux types de propriétés distinctes :

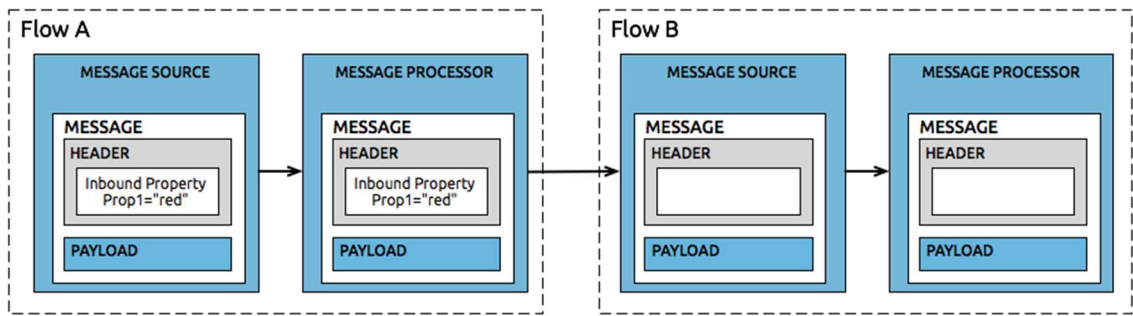
Inbound : Ces propriétés peuvent être accédées mais pas modifiées par l'utilisateur. Elles assurent la cohérence du message au travers des processeurs jusqu'à la fin du flux en cours. Ces propriétés ne sont pas transmises entre flux.

Outbound : Ces propriétés sont manipulables par l'utilisateur et peuvent se transmettre d'un flux à un autre. En fonction du type de transport, les propriétés *Outbound* peuvent se convertir en propriété *Inbound* ou rester dans leur état initial.

Les images ci-dessous décrivent ce processus :

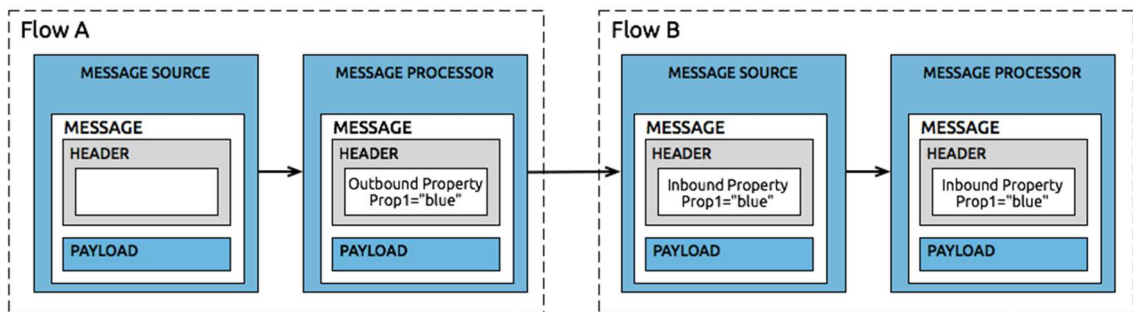
² Voir chapitre 10 - Glossaire

Figure 4 : Transport de propriétés Inbound



<http://www.mulesoft.org/documentation/display/current/Mule+Message+Structure>

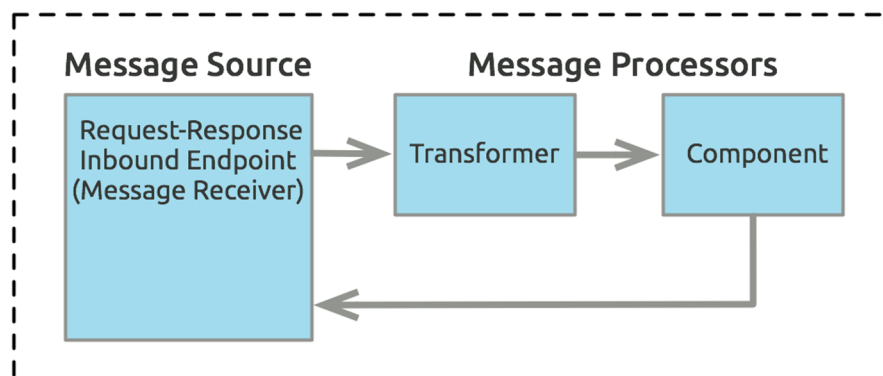
Figure 5 : Transport de propriétés Outbound



<http://www.mulesoft.org/documentation/display/current/Mule+Message+Structure>

3.3 Éléments du flux

Figure 6 : Schéma général d'un flux



<http://www.mulesoft.org/documentation/display/current/Mule+Concepts>

3.3.1 Connecteurs

Les connecteurs (Connector) sont des composants qui permettent de créer des applications Mule avec les API d'autres applications externes. C'est généralement d'un composant connecteur que l'on obtient la source de notre message. Quelques connecteurs les plus utilisés :

File : Surveille un dossier et active un évènement dès qu'un fichier y est déposé. Il permet également de lire et d'écrire un fichier depuis une application Mule.

HTTP : Effectue une requête http, en one-way ou request-response.

Database : Permet la connexion à une base de données, ainsi que l'exécution de requêtes sur celle-ci.

Web Service Consumer : Permet de faire appel à un web service.

Dans les applications Mule mises en place à la RTS, c'est surtout le composant *File* qui sera utilisé afin de détecter l'arrivée d'un fichier XML dans un dossier surveillé et de l'acheminer au bon endroit.

3.3.2 Transformateurs

Les transformateurs (Transformer) sont des composants chargés de transformer le *Payload* de et vers différents types. Quelques transformateurs utilisés fréquemment :

File To String : Convertit un objet File (Stream) en objet String, ce qui en permet la manipulation à l'intérieur du flux.

Variable : Permet de définir une variable qui sera ajoutée au message.

Expression : Permet d'évaluer des expressions à l'intérieur du message.

Set Payload : Définit un nouveau payload.

XSLT : Applique une transformation XSLT sur le payload.

Il existe beaucoup de transformateurs qui permettent la conversion du message d'un format à un autre, comme *File To String*, *DOM To XML*, *Object To JSON* etc. Dans l'implémentation des applications RTS, ce sont les transformateurs *FileToString* et *Variable* qui sont les plus utilisés.

C'est également dans cette catégorie que l'on peut trouver un des composants phare de Mule : le *DataMapper*. Ce transformateur propose de convertir une source de donnée en une autre très facilement. Les formats d'entrée et de sortie sont très nombreux. Par exemple, passer d'un format CSV à XML, JSON ou autre. Ce transformateur fait partie des composants payants.

3.3.3 Composants

Les composants (Component) représentent la logique business de l'application Mule. Ils prennent en général le payload, effectuent les traitements puis retournent un nouvel objet qui sera le nouveau *Payload*. Parmi les composants, on trouvera notamment :

Java : Permet d'inclure un traitement Java.

Script : Permet d'inclure un script

Logger : Permet d'afficher dans la console, sans altérer le payload, diverses informations.

Batch Execute : Permet d'exécuter un fichier batch

Les composants sont essentiels à la réalisation d'applications complexes.

3.3.4 Étendues

Les étendues (Scope) sont des blocks de traitement qui doivent être placés dans le composant *Flow*. Certains n'acceptent qu'un composant à l'intérieur (*Poll* par exemple) mais proposent en contrepartie certaines fonctionnalités comme le déclenchement périodique. D'autres, comme le *Sub-Flow*, permettent de mettre plus d'un block dans le processeur de messages. Les étendues les plus courantes sont :

Flow : Conteneur général des applications.

Sub-Flow : Sous-conteneur, majoritairement référencé dans les composants *Flow-Ref*

Poll : Permet de déclencher périodiquement un événement et/ou un composant placé à l'intérieur.

Composite Source : Permet de définir plusieurs sources pour le message.

Toutes les applications développées intègrent les composants dans une étendue *Flow* (et éventuellement *Sub-Flow* pour les petits traitements), y compris pour l'étendue *Poll*.

3.3.5 Filtres

Les filtres (Filter) permettent de ne faire passer que certains messages, répondant à des critères spécifiques, à la prochaine étape du flux. Par exemple, on peut filtrer sur le contenu du message ou le nom du fichier. Quelques exemples :

Expression : Permet de filtrer via une expression Mule.

Payload : Filtre selon le type de payload.

Regex : Filtre avec une expression régulière.

And / Or / Not : Opérateurs logiques classiques.

Dans les applications RTS, très peu de filtres sont mis en place car on travaille sur des fichiers de métadonnées ; tous doivent être traités et les fichiers sont considérés comme valides au moment de leur entrée dans le flux. Les filtres peuvent également

être intégrés directement aux propriétés de la plupart des composants. Une exception pour le prototype « MassDL » (voir 5.10 ci-dessous) qui utilise un filtre afin de valider le format d'un fichier XML en appelant un fichier XSD.

3.3.6 Contrôles de flux

Les contrôleurs de flux (Flow Control) permettent de router le message selon des critères prédéfinis ainsi que d'effectuer des opérations sur les collections. Dans cette section, il y a :

Choice : Ce composant correspond à un routeur classique, ou à l'équivalent Mule du Switch de Java. Il permet de définir différents chemins avec une condition de garde pour chacun d'eux, ainsi qu'une route par défaut.

Scatter-gather : Ce composant envoie le même message sur chaque branche. Il permet ainsi d'avoir plusieurs chemins parallèles avec la même source.

First Successful : Envoie le message vers la première branche à être en mesure de l'accepter. Par exemple, pour la communication avec des serveurs.

Round Robin : Transmet chaque message au prochain processeur de message selon une liste circulaire.

3.3.7 Gestion des erreurs

Dans Mule, il existe une petite collection de composants qui permettent de définir la façon dont vont être traitées les erreurs. Ainsi, il est possible de définir pour chaque flux (mais pas pour les sous flux !) une gestion des erreurs. Il est également possible de définir une gestion globale. Dans ces composants de gestion d'erreurs, il est tout à fait possible d'utiliser les composants standards tels que le *File* pour, par exemple, écrire le fichier ayant provoqué l'erreur dans un dossier spécifique ou encore de générer un fichier de log. Les composants principaux sont :

Catch Exception Strategy : Composant permettant de créer sa gestion d'erreurs.

Reference Exception Strategy : Permet de référencer une stratégie de gestion.

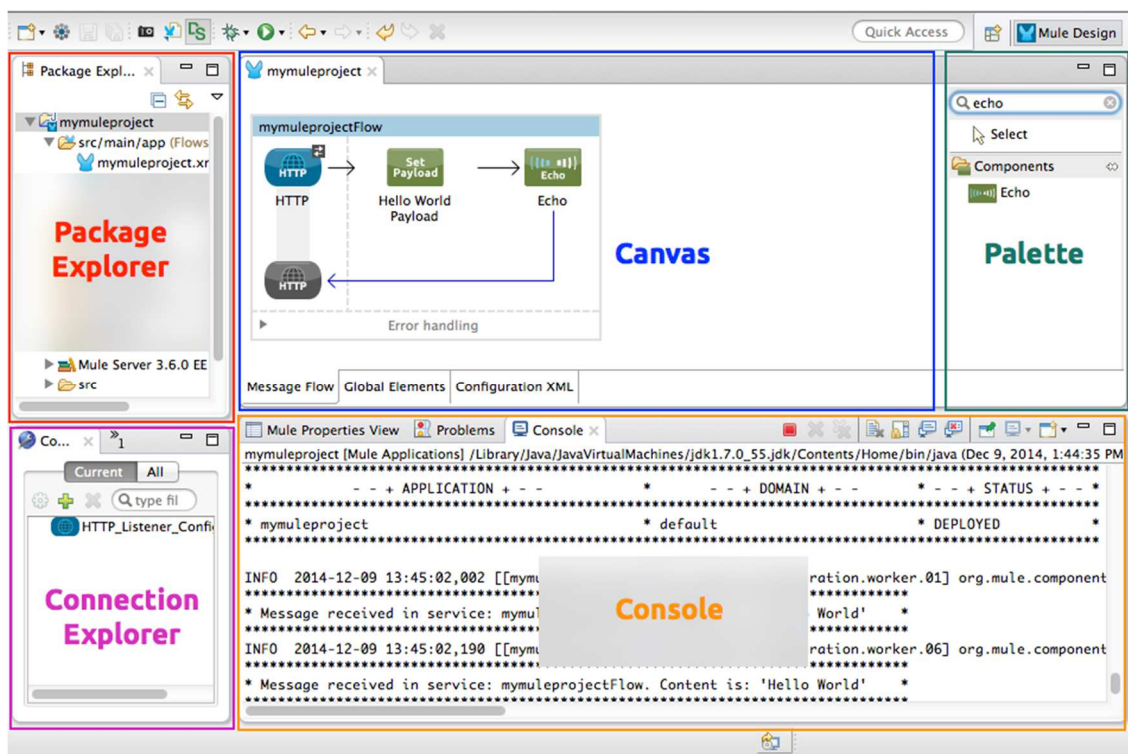
3.4 Les nouveautés

La version actuellement déployée à la RTS est la version 2.5. La nouvelle version, 3.6.1, propose quelques fonctionnalités supplémentaires dont je vais présenter une partie. Étant donné le grand nombre de fonctions ajoutées / supprimées / modifiées, la liste n'est pas exhaustive.

3.4.1 Anypoint Studio

Il s'agit sans doute du changement le plus majeur de l'application. Il devient possible de définir ses flux non plus par du code exclusivement XML mais également via une interface graphique intégrée à Eclipse³ et qui permet, via le drag-and-drop, de créer rapidement et efficacement le schéma général de l'application. De plus, cette interface offre de nombreuses options de création et de debugge. Elle offre également la possibilité de changer de runtime (Community / Enterprise) très facilement. L'image ci-dessous décrit l'interface de Mule, divisée en 5 parties distinctes.

Figure 7 : Interface Anypoint Studio



<http://www.mulesoft.org/documentation/display/current/Anypoint+Studio+Essentials>

Le *Package Explorer* : Montre les fichiers et les dossiers du projet sous la forme d'une arborescence que l'on peut étendre ou contracter.

Le *Canvas* : Propose un espace blanc où l'on peut déposer les composants de l'application. C'est également à cet endroit que l'on peut naviguer entre les différentes vues (message flow / configuration XML) de l'application. Les vues sont expliquées plus bas dans ce document.

La *Palette* : Affiche les composants à disposer pour créer l'application.

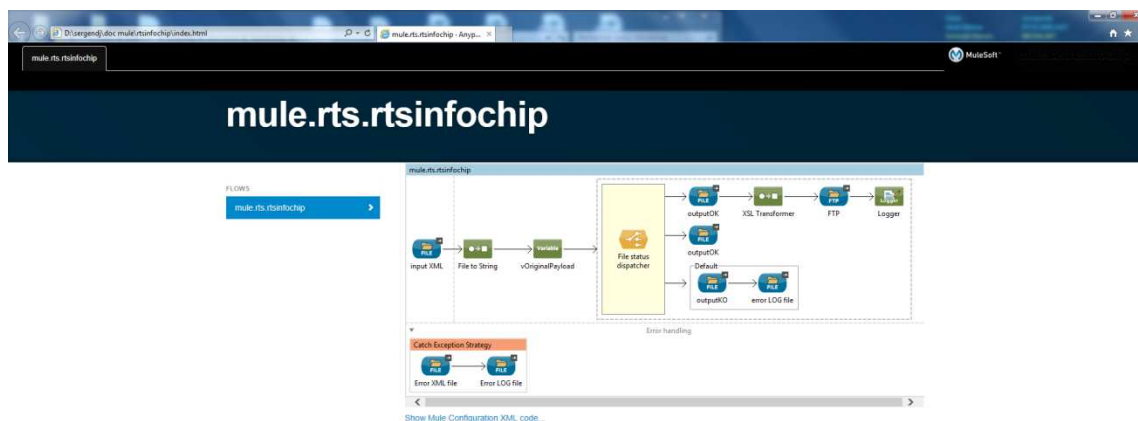
³ Environnement de développement, voir chapitre 10 - Glossaire

Le *Connection Explorer* : Cet élément affiche les différents connecteurs qui ont une portée globale pour un accès rapide à leur configuration.

La *Console* : C'est dans la console que l'on va pouvoir suivre notre flux, le debugger etc. Une partie des composants affichent des informations lorsqu'un message les traverse. C'est dans cette zone que le composant *Logger* pourra afficher les informations.

Anypoint Studio intègre également d'autres fonctionnalités comme le debugge, la génération de documentation au format HTML, l'export des flux sous forme d'image, des perspectives particulières pour le développement etc.

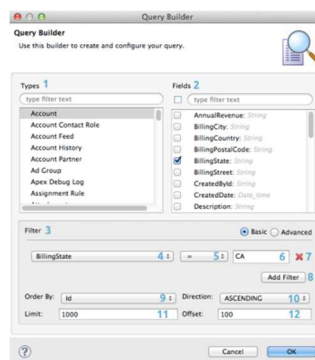
Figure 8 : Documentation automatique générée par Mule



3.4.2 DataSense Language Query

Il s'agit d'un langage de requête ressemblant à SQL. Il permet de créer des requêtes et de retrouver des métadonnées d'applications sans avoir à fournir d'effort manuel pour découvrir et comprendre quels sont les champs utilisables et questionnables. Ce nouvel outil est lié à un Query Builder, qui permet de créer rapidement et efficacement les requêtes qui seront par la suite utilisées dans d'autres composants.

Figure 9 : Interface Query Builder



<http://www.mulesoft.org/documentation/display/current/DataSense+Query+Editor>

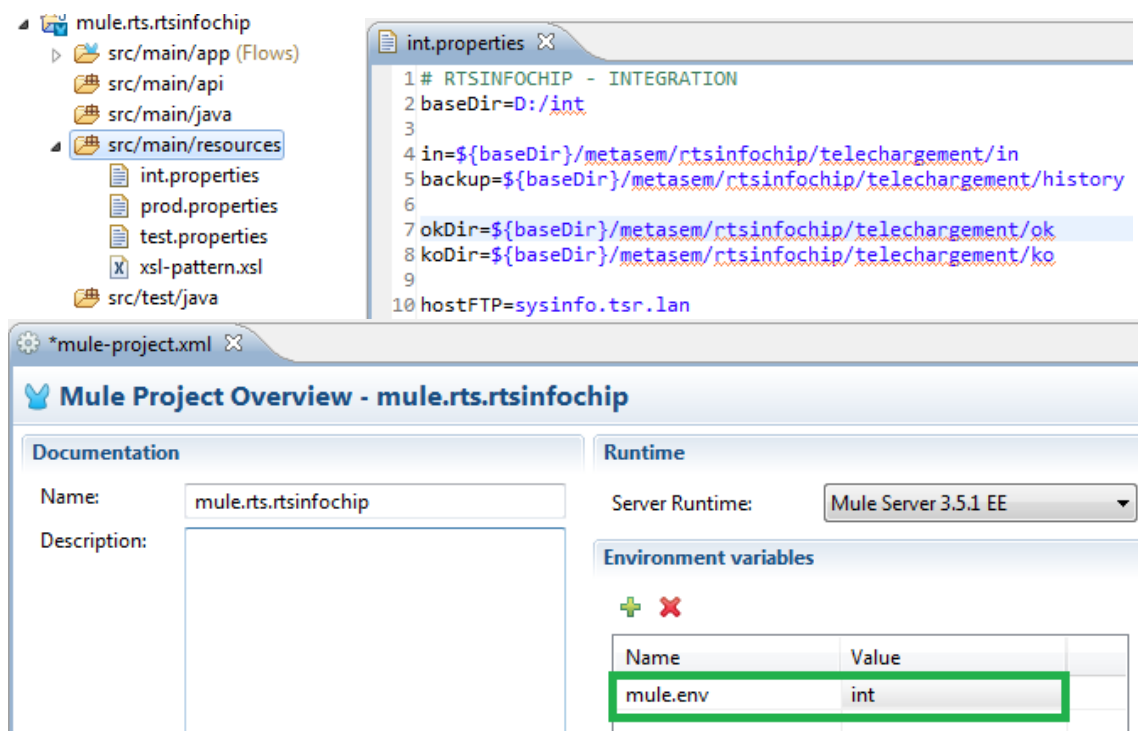
3.4.3 Project Environment Settings

Cette fonction clé permet de travailler sur de multiples environnements avec une seule application Mule. Pour faire ceci, 3 étapes simples suffisent. Il faut :

- Créer un fichier de propriété par environnement (exemple : *prod.properties* ; *test.properties*, *int.properties*) dans l'application.
- Définir un objet global *Property Placeholder* qui fera le lien entre l'environnement et le fichier qu'il faut charger.
- Définir la variable d'environnement. Elle peut donc prendre 3 valeurs : *prod*, *test*, *int*.

La procédure employée pour mettre en place ce système est précisée plus bas.

Figure 10 : Gestion d'environnements multiples



3.4.4 CloudHub Sandbox

Les Sandbox nouvellement disponibles dans la version CloudHub de l'application permettent de gérer efficacement les environnements de test et d'intégration sans altérer le bon fonctionnement de l'application. Ces environnements restrictifs empêchent les mauvaises manipulations de toucher l'environnement de production.

3.4.5 Watermarks

Permet une gestion plus fine des données dans l'élément *Poll*. Par exemple, si l'on veut charger uniquement les nouveaux éléments, quel que soit le contexte, on peut faire appel aux watermarks qui vont garder une trace du dernier objet à avoir été pris en charge ou l'heure à laquelle cela s'est produit.

3.4.6 Prochaine version

Note : la version 3.6.1 est sortie durant la rédaction de ce document. Ces nouveautés ont donc été intégrées aux prototypes.

La prochaine version (3.6.1) va inclure :

- Une mise à jour des performances du composant HTTP
- Un connecteur officiel AMQP⁴, prenant en charge plus de fonctions que la version communautaire actuellement disponible.

3.5 Langages et développement

Pour fonctionner, Mule utilise plusieurs langages qui, une fois combinés, permettent de créer des applications. Du langage de programmation Java au langage d'expression MEL⁵ en passant par le XSLT⁶, les possibilités sont nombreuses.

3.5.1 XML – XSLT

Dans Mule, chaque application est représentée sous la forme d'un fichier XML décrivant les différents composants du programme. En dehors du fait qu'il utilise XML pour définir l'application et qu'il dispose donc d'un éditeur, Mule manipule également très bien ce genre de données. Toute une série de transformateurs permettent la prise en charge de ce format, sa conversion et son exploitation. Le XSLT, qui permet d'appliquer une méthode de transformation particulière à un fichier XML est également très bien pris en charge et permet d'effectuer facilement ce genre d'opérations. Il faut cependant noter que XSLT n'est pas un langage de programmation à proprement parler, mais un langage de transformations. Un fichier XSL est représenté sous la forme d'un fichier XML et se base sur le positionnement avec XPATH⁷. Un fichier XSL doit obligatoirement être accompagné d'un fichier XML pour fonctionner. Parmi les subtilités du langage, on notera par exemple les variables qui se comportent en constantes une fois initialisées.

3.5.2 Java

Le logiciel Mule est codé en Java et son interface utilisateur est intégrée à Eclipse, le célèbre environnement de production de logiciels libre. Il permet d'intégrer très facilement du code Java à l'intérieur d'un flux, soit par l'intermédiaire d'un composant Java qui appelle une classe soit directement à l'intérieur des variables et des

⁴ Advanced Message Queuing Protocol, voir chapitre 10 - Glossaire

⁵ Mule Expression Language, voir chapitre 10 - Glossaire

⁶ eXtensible Stylesheet Language Transformations, voir chapitre 10 - Glossaire

⁷ Langage de positionnement dans un fichier XML, voir chapitre 10 - Glossaire

expressions. On pourrait par exemple créer une classe qui retourne le nom standardisé d'un fichier, et l'appeler directement depuis une variable en utilisant le langage MEL.

3.5.3 MEL

Mule Expression Language est un langage d'évaluation d'expressions spécifique à Mule. Il permet d'accéder et d'évaluer les données d'un message Mule. Il peut également servir à filtrer, router ou effectuer d'autres actions sur les flux, les messages etc. L'avantage de cette méthode, c'est la standardisation de l'accès aux informations dans tous les composants de Mule et les nombreuses possibilités d'implémentation. Le MEL peut s'utiliser presque partout.

3.5.4 Expressions régulières

A bien des niveaux, les expressions régulières sont très présentes dans Mule. On peut les utiliser, par exemple, pour filtrer les messages, le nom des fichiers entrant, etc.

3.5.5 Fichiers « .properties »

Ces fichiers, créés à la main, permettent de gérer les propriétés de l'application. De ce fait, ils offrent également la possibilité de gérer les différences entre les environnements. Lorsque l'on veut changer d'environnement, il suffit de changer la valeur de la variable d'environnement correspondante et de relancer l'application. Les nouveaux paramètres sont chargés et l'application est prête à travailler. Cette technique est très puissante puisqu'elle permet de développer, avec une seule application, une multitude d'environnements en parallèle sans aucune contrainte. De plus, elle permet de déployer facilement l'application sur l'environnement désiré à l'aide d'une ligne de commande. Il faudra néanmoins procéder à quelques ajustements lors de la mise en production (voir chapitre 6.1 ci-dessous).

3.5.6 Autres possibilités

Le logiciel Mule est conçu pour permettre les échanges et les communications entre des systèmes hétérogènes. C'est pourquoi il ne se limite pas aux quelques langages cités ci-dessus. Il met à disposition une série de transformateurs et de composants qui, du Python jusqu'au JavaScript, permettent d'inclure une multitude de langages au flux. Tout cela combiné permet de créer des applications très diverses.

4. Modélisation des flux

Chaque flux doit disposer d'un modèle afin de pouvoir communiquer à son sujet avec le plus grand nombre de gens possible. Il s'agit de l'une des contraintes de documentation liées au projet. Dans ce chapitre, la méthode de modélisation et les outils sont décrits.

4.1 Principe de modélisation standardisée

Afin de standardiser la documentation relative aux processus de la RTS, les flux seront modélisés en UML, au travers de diagrammes d'activité⁸. Les diagrammes sont prévus pour être orientés haut niveau plutôt que technique, permettant de définir le contexte général et les entrées/sorties de l'application.

Chaque application est définie dans un fichier au format XML. Ce code XML contient chaque composant, les références aux fichiers de configuration etc. Il est accessible et modifiable via la vue *Configuration XML*.

La vue *Message Flow* (voir chapitre 4.3 ci-dessous) du logiciel nous propose une représentation visuelle du fichier XML qui définit l'application. Le code XML et le modèle Mule ne sont donc que deux façons différentes de visualiser un seul élément. Il est par conséquent possible d'alterner entre la vue classique en XML et la vue graphique afin de réaliser l'application. La vue *Message Flow* est une des nouveautés de Mule 3.5 et propose de définir rapidement et efficacement, à l'aide de composants que l'on glisse et que l'on dépose, le schéma général de l'application. Dans ce rapport, ces modèles sont utilisés afin de détailler le fonctionnement des prototypes.

4.2 Outils et logiciels

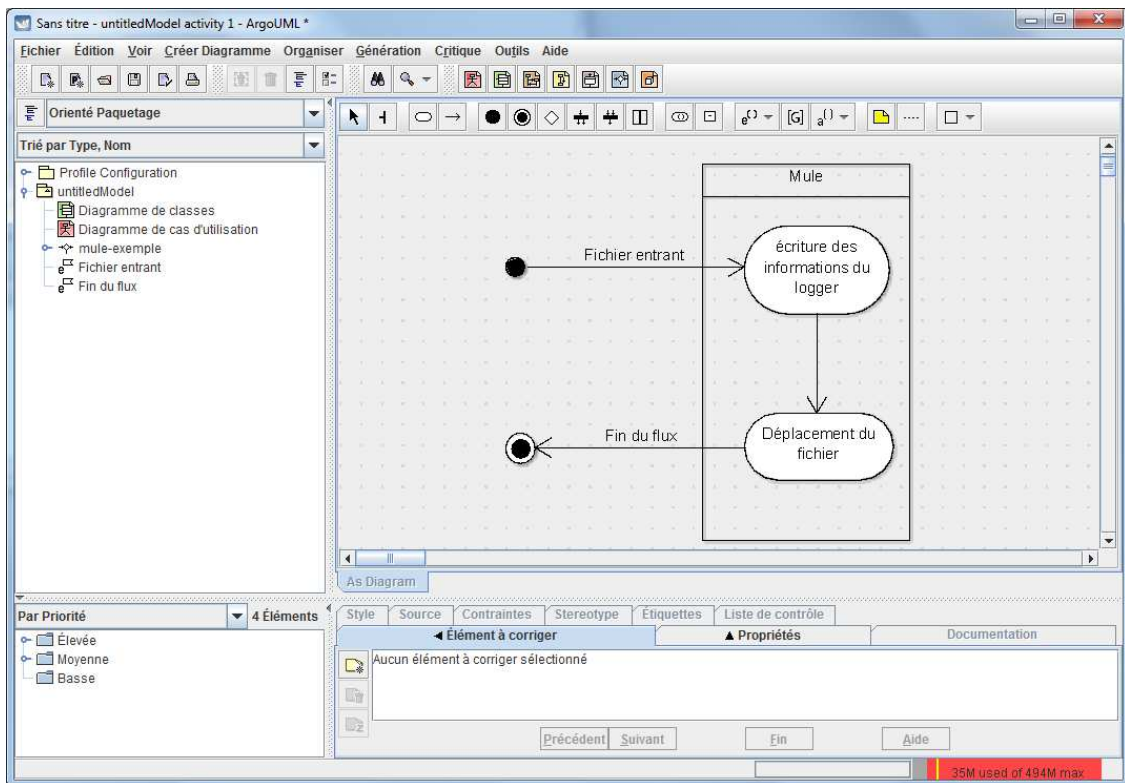
Il s'agit ici de décrire les outils utilisés pour modéliser les processus. Le choix doit se porter sur des logiciels libres, l'offre dans ce domaine pouvant pleinement répondre aux besoins actuels.

4.2.1 ArgoUML

Ce logiciel de création UML nous permet de réaliser simplement et rapidement différents types de diagrammes. Il est open source, et sera utilisé afin de réaliser chaque diagramme d'activité du projet. Il est également légèrement instable.

⁸ Diagramme représentant le Workflow, voir chapitre 10 - Glossaire

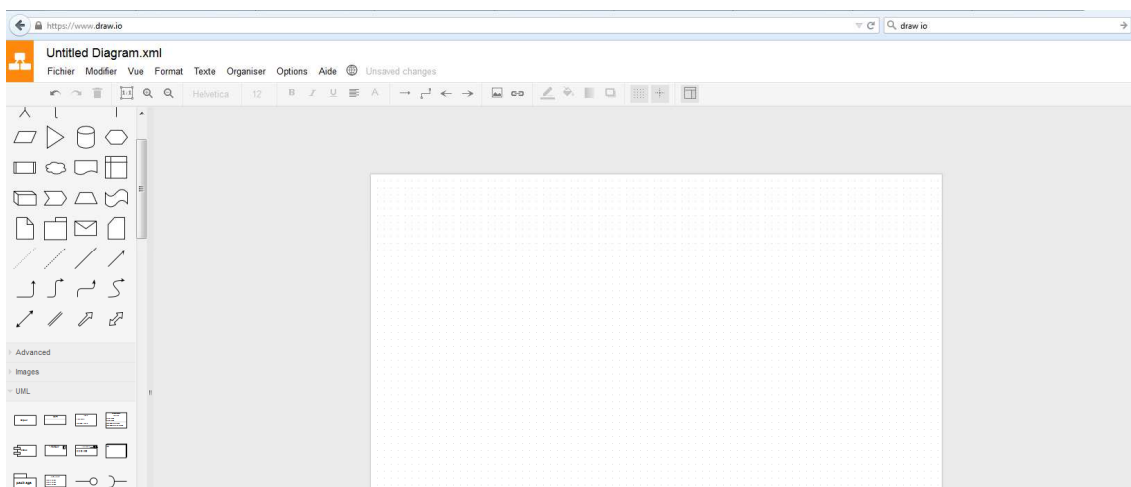
Figure 11 : Interface ArgoUML



4.2.2 Draw IO

Ce service est disponible en ligne (<https://www.draw.io>) et propose de créer des diagrammes et modèles gratuitement. C'est une alternative, du point de vue de la RTS, à la modélisation par ArgoUML. Cependant, durant le projet, il est prévu de favoriser le logiciel UML, spécialisé, plutôt que le service en ligne plus général.

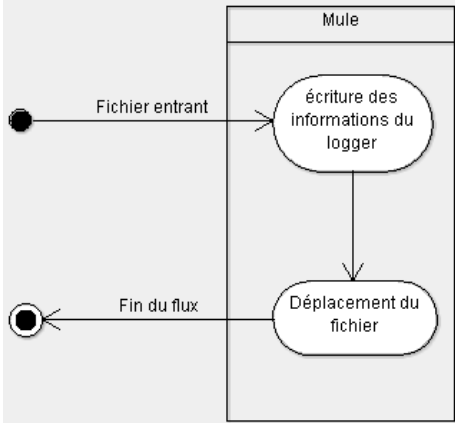
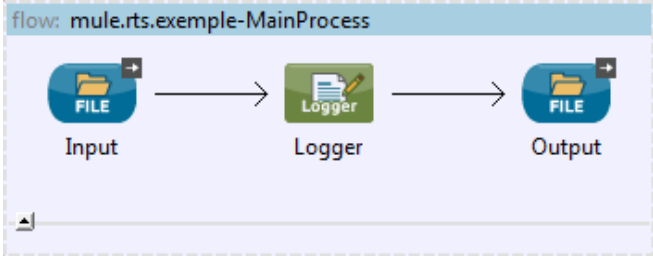
Figure 12 : Interface Draw IO



4.3 Exemples de modèles

Prenons une application simple : un fichier entre dans un dossier surveillé (Source), est copié dans un dossier de backup, un logger affiche dans la console le nom du fichier (Message Processor) et celui-ci est déplacé dans un dossier de sortie. Aucune manipulation du fichier n'est effectuée durant ce flux.

Tableau 2 : Représentation des flux

<p>UML</p>	<p>Figure 13 : Exemple de modélisation UML</p> 
<p>MULE</p>	<p>Figure 14 : Exemple de vue « Message Flow »</p> 
<p>XML</p>	<p>Figure 15 : Exemple de vue « Configuration XML »</p> <pre> 1 <?xml version="1.0" encoding="UTF-8"?> 2 3 <mule xmlns:tracking="http://www.mulesoft.org/schema/mule/ee/tracking" xmlns:context="http://www.springframework.org/schema/context" 4 xmlns:file="http://www.mulesoft.org/schema/mule/file" xmlns="http://www.mulesoft.org/schema/mule/core" 5 xmlns:doc="http://www.mulesoft.org/schema/mule/documentation" 6 xmlns:spring="http://www.springframework.org/schema/beans" version="EE-3.5.1" 7 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" 8 xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-current.xsd 9 http://www.mulesoft.org/schema/mule/core http://www.mulesoft.org/schema/mule/core/current/mule.xsd 10 http://www.mulesoft.org/schema/mule/file http://www.mulesoft.org/schema/mule/file/current/mule-file.xsd 11 http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-current.xsd 12 http://www.mulesoft.org/schema/mule/ee/tracking http://www.mulesoft.org/schema/mule/ee/tracking/current/mule-tracking-ee.xsd"> 13 14 <context:property-placeholder location="\${mule.env}.properties" /> 15 16 <flow name="mule.rts.exemple-MainProcess" doc:name="mule.rts.exemple-MainProcess"> 17 <file:inbound-endpoint path="\${input}" 18 moveToPattern="#[message.inboundProperties.'originalFilename']" 19 moveToDirectory="\${backup}" responseTimeout="10000" doc:name="Input" /> 20 <logger level="INFO" doc:name="Logger" /> 21 <file:outbound-endpoint path="\${output}" 22 outputPattern="#[message.inboundProperties.'originalFilename']" 23 responseTimeout="10000" doc:name="Output" /> 24 </flow> 25 </mule> 26 </pre>

5. Prototypes

Les prototypes sont décrits de la façon suivante :

En premier lieu, une description des éléments communs à tous les flux. Ensuite, pour chaque prototype, une description du contexte dans lequel il s'inscrit ainsi qu'un résumé de son fonctionnement, de ses contraintes et des préconditions non testées. A la suite de cela, deux modèles sont ajoutés. L'un, représenté sous la forme d'un diagramme d'activité, décrit le workflow avec une vision de haut niveau. Le deuxième, relatif à Mule, représente le flux dans la version graphique fournie par le logiciel. Un exemple des entrées/sorties est présenté et, enfin, la fonction de chaque composant non générique du flux est décrite. Pour finir, le prototype est rapidement analysé selon deux axes : la réussite de la mise en place de la fonction désirée ainsi que la complexité de création et d'analyse. Enfin, sont décrits les points positifs et négatifs qui peuvent représenter une fonction, une compétence acquise, une configuration particulière ou encore une optimisation par rapport à une utilisation précédente.

Les éléments variables sont signalés entre accolades : **{variable}**.

Les éléments alternatifs sont signalés entre crochets : **[choix 1 | choix 2]**.

Les prototypes sont développés pour les environnements : **dev, int, test, prod**.

Les tests effectués pour valider les prototypes sont décrits plus loin dans ce document, dans la section Evaluation des prototypes. Tous les prototypes possèdent une gestion d'erreurs, de logs, sont compilables et gèrent un historique de tous les fichiers passés au travers. La perte de fichiers est ainsi écartée et les fichiers en erreur sont traités par un technicien.

5.1 Éléments génériques

Les composants ci-dessous ont un comportement majoritairement similaire quel que soit le flux. Ils ne seront ainsi détaillés dans les prototypes que si leurs actions sont importantes à la compréhension globale du flux ou si celles-ci diffèrent grandement de leur utilisation d'origine.

Composant	Rôle
Scope Flow	Englobe (en général) la totalité des composants de l'application. Il s'agit d'un composant utilisé dans chaque flux.
Scope Sub-Flow	Englobe un sous processus ou une partie de l'application. On y accède via le composant <i>Flow Ref</i> .
Scope Poll	Exécute un appel à fréquence régulière, par exemple sur une base de données.
Component Flow-Ref	Pointe un flux ou un sous-flux depuis un autre.
Component Logger	Affiche des informations dans la console.
Connector File	En input, prend un fichier du dossier désigné. Peut également en faire une copie de backup. En output, va écrire un fichier à l'emplacement désigné avec le nom défini.
Transformers	FileToString : Convertit un fichier en String DOMToXML : Convertit un Document (p.e un stream XML) en XML XSLT : Applique une transformation XSLT à un fichier XML
Variables	Contient une variable qui peut être définie par un appel java, une expression MEL ⁹ , du texte etc.

⁹ Mule Expression Language, voir chapitre 10 - Glossaire

5.2 N°1 : Rtsinfochip

Nom du flux	mule.rts.rtsinfochip
Contexte	Lorsqu'un utilisateur (un journaliste, documentaliste...) souhaite obtenir de la matière (par exemple une vidéo), il effectue sa demande via une application dédiée. Cette application génère une fiche XML standard qui est déposée dans le dossier surveillé (input) de l'application Mule rtsinfochip. L'application va analyser ce fichier et l'acheminer selon son statut d'avancement. Cela aura pour effet de mettre ce fichier à disposition de la chaîne IP ¹⁰ .
Description	<p>Déclencheur : Fichier entrant dans le dossier surveillé.</p> <p>Processus : Déplacement et transformation de fiche XML.</p> <p>Résultat : Fichier transformé sur FTP.</p> <p>L'application surveille le point d'entrée puis, dès qu'un fichier XML entre, le convertit en String et le copie dans le dossier de backup. Le routeur vérifie alors le contenu de la balise <status> à l'aide d'une expression XPATH¹¹. Si le <status> est de type « running », le fichier est déplacé dans le dossier OK de l'application afin de conserver un historique. Si le <status> est de type « success », le fichier est copié dans le dossier OK (backup), puis une transformation XSLT lui est appliquée. Le nouveau document XML est alors envoyé sur un serveur FTP et le Logger affiche différentes informations. Tous les autres fichiers sont envoyés dans le dossier KO de l'application, où un outil de monitoring PHP s'occupera de lever une erreur afin de prévenir les techniciens.</p>
Contraintes	<p>Le fichier ne doit pas être perdu en cas d'arrêt du processus.</p> <p>Traitements simultanés : 1 (processus synchrone)</p>
Préconditions	Le fichier XML reçu est considéré valide.

¹⁰ Diffuse les médias sur le site web de la RTS

¹¹ Langage de positionnement dans un fichier XML, voir chapitre 10 - Glossaire

Modèles

Figure 16 : Modèle UML Rtsinfochip

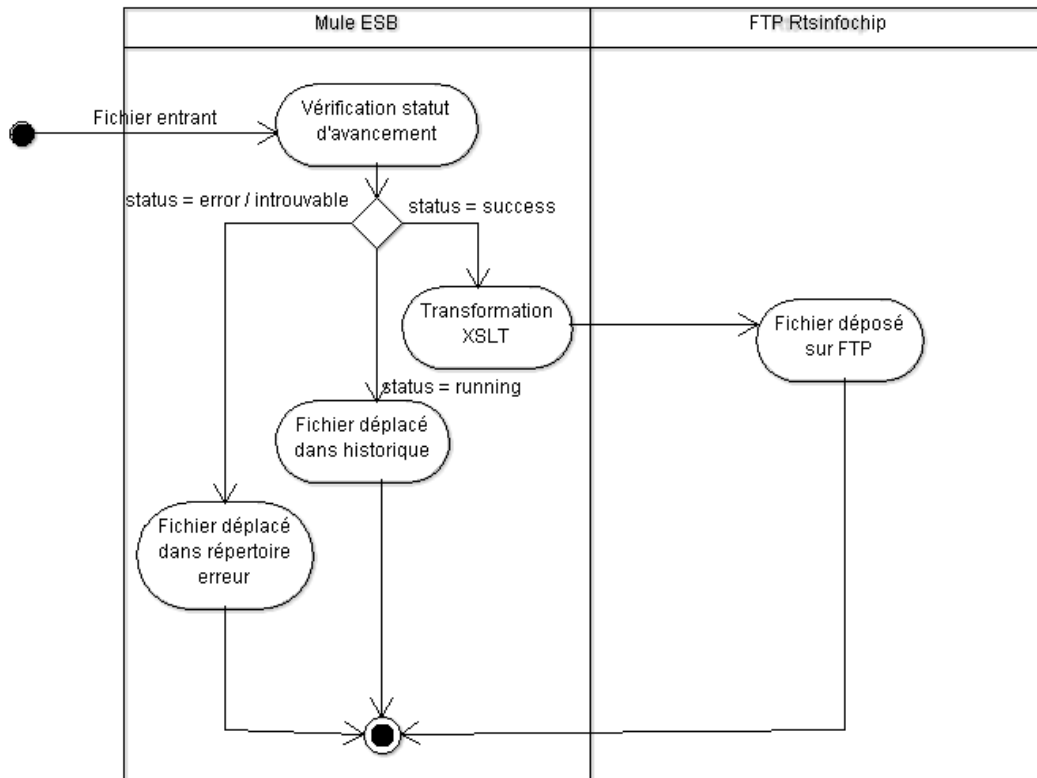
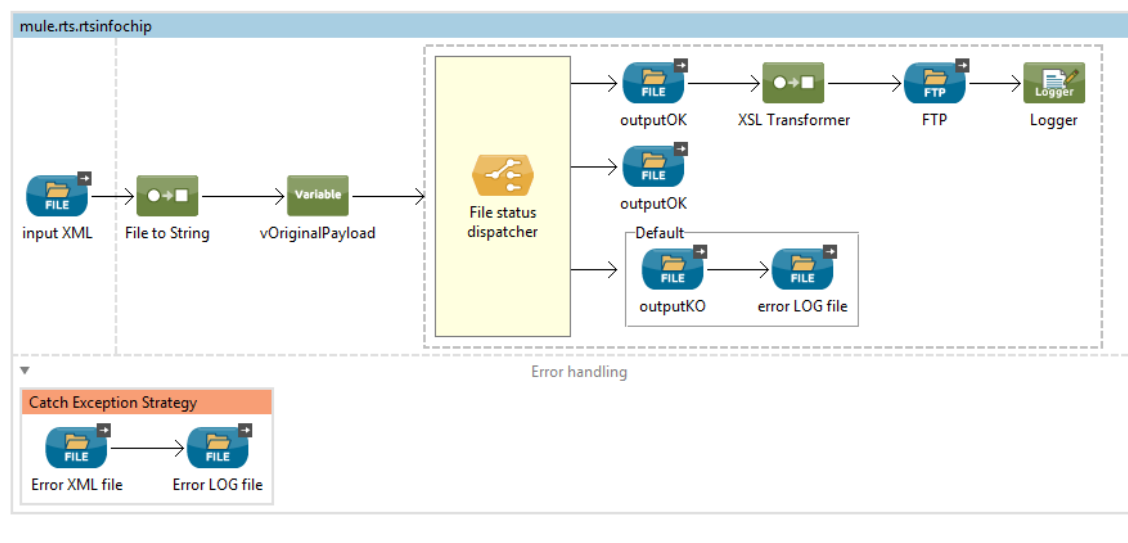


Figure 17 : Modèle Mule Rtsinfochip



Entrée /metasem/rtsinfochip/telechargement/in

```
<?xml version="1.0" encoding="ISO-8859-1"?><Demande_Restitution>
  <Request_Id>REQ-358177</Request_Id>
  <Request_Media_Item_Id>RMI-307444</Request_Media_Item_Id>
  <Request_Action>restitution</Request_Action>
  <Status>success</Status>
  <Error_Message/>
  <Description>280115_BDL_369</Description>
  <Creator>Milesi, Baptiste Daniel (RTS)</Creator>
  <Notification_Date>26.01.2015 10:11</Notification_Date>
  <Media_Dest>ZM011960.mxf</Media_Dest>
  <Tc_Duration>00:00:38:08</Tc_Duration>
  <Media_Src>ZM011960.mxf</Media_Src>
  <Title>280115_BDL_369 - Autopromotion</Title>
</Demande_Restitution>
```

Sortie FTP : sysinfo.tsr.lan

```
<ArchiveList>
  <Title>280115_BDL_369 - Autopromotion</Title>
  <MediaID>ZM011960.mxf</MediaID>
  <Comments>280115_BDL_369</Comments>
  <REQ>REQ-358177</REQ><RMI>RMI-307444</RMI>
  <STATUS/>
  <Archivist>Milesi, Baptiste Daniel (RTS)</Archivist>
</ArchiveList>
```

X	Type Operation	Capa Limit	En Cours	En Attente	Statut
✘	mediaAnalyzer:thumbFile	20	4	0	WARNING_old ERROR_old
✔	sparkExchange:archive	20	1	0	
✔	sparkExchange:restauration	20	3	0	
✔	telestream:transcodeFile	16	2	0	

✘ Automate workers errors (II) [Liste détaillée](#)

Type Operation	Nombre d'erreurs
sparkExchange:restauration	11

27-04-2015 11:12:34

Composant	Nom	Rôle
Connector FTP	FTP sysinfo	Dépose le fichier de sortie sur le serveur FTP.
Flow Control Choice	file status dispatcher	Vérifie le contenu de la balise <status> du fichier input avec XPATH et route le message en fonction de la valeur (« success » / « running » / autre).

Analyse du prototype

Ce premier flux, ne pose aucun problème technique dans sa mise en place. Il aura permis de se familiariser avec le connecteur *File* utilisé dans la plupart des prototypes ainsi que d'appréhender l'utilisation du composant FTP et les transformations XSLT.

- | | |
|---|--|
| <ul style="list-style-type: none">+ Prise en main du connecteur File+ Transformation XSLT+ Intégration XPATH dans MEL | <ul style="list-style-type: none">- Composant FTP différent entre la version Entreprise et Community |
|---|--|

5.3 N°2 : Thumbfile Dispatch

Nom du flux	mule.rts.thumbfile
Contexte	Les applications MediaAnalyzer, situés sur 3 serveurs différents, génèrent des vignettes pour chaque média afin de faciliter le parcours des vidéos (une vignette = un changement de plan). Le flux doit répartir la charge entre chaque serveur, tout en respectant le nombre maximum de fichiers dans chaque dossier.
Description	<p>Déclencheur : Fichier entrant dans le dossier surveillé.</p> <p>Processus : Répartition de charge entre applications parallèles.</p> <p>Résultat : Fichier déplacé.</p> <hr/> <p>L'application surveille le point d'entrée puis, dès qu'un fichier entre, le convertit en String et initialise une variable vDirectoryTarget en appelant une méthode d'une classe Java. Cette procédure va chercher dans les propriétés de l'application les dossiers cibles potentiels et compte le nombre de fichiers présents dans chacun de ces dossiers afin de répartir la charge. Si aucun dossier n'est disponible, la fonction Java attend 5 secondes avant de réessayer. Lorsqu'un dossier est défini, le fichier y est écrit.</p>
Contraintes	<p>Traitements simultanés : 1 (processus synchrone)</p> <p>Maximum de fichiers par dossier : 5</p> <p>Nombre de dossiers possibles : 3</p> <p>Nombre d'essais possibles : illimités</p>
Préconditions	Le fichier reçu est considéré valide.

Figure 18 : Modèle UML Thumbfile Dispatch

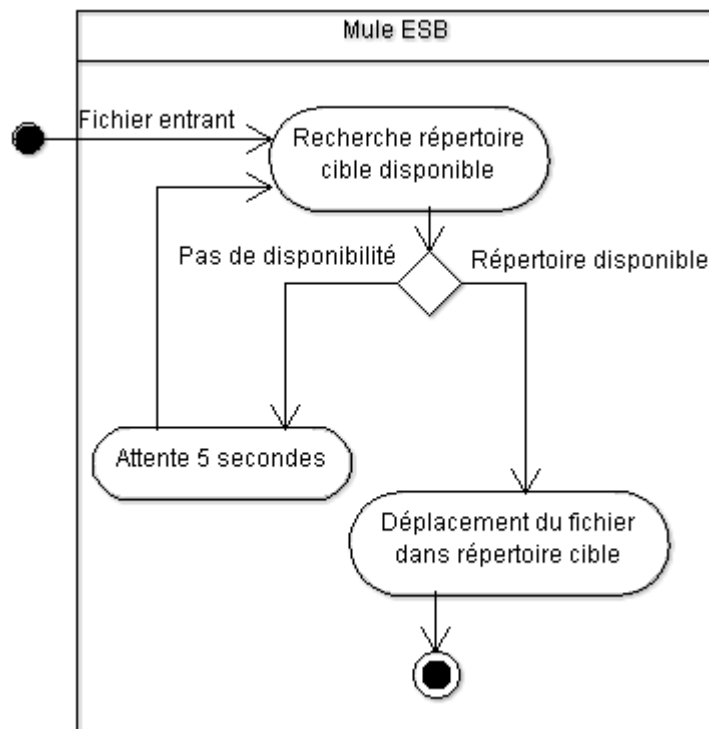
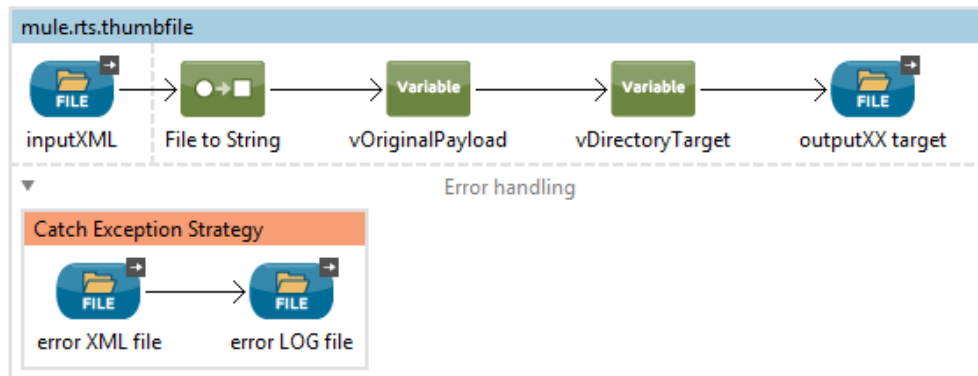


Figure 19 : Modèle Mule Thumbfile Dispatch



Entrée /metasem/dam/thumb/in		
Fichier XML dans le dossier de temporisation.		
Sortie /metasem/dam/thumb/[in01 in02 in03]		
Fichier déplacé dans le dossier d'output défini.		
Composant	Nom	Rôle
Transformers Variable	vDirectoryTarget	Variable qui contient le dossier de destination, renvoyé par une méthode Java de la classe <i>FileCounter</i> .
	vOriginalPayload	Contient le payload d'entrée pour des raisons de backup (transformateur global, voir 6.3 ci-dessous).
Analyse du prototype		
Ce deuxième prototype a permis de comprendre l'interaction entre Mule et les traitements Java au travers des variables. Ici, la logique métier est entièrement gérée depuis la variable. Celle-ci appelle un algorithme qui va à la fois vérifier la disponibilité des dossiers de destination, temporiser et répartir la charge.		
<ul style="list-style-type: none"> + Utilisation de variables pour les traitements java + Répartition de charge + Définition dynamique d'un point de sortie + Traitements synchrones pour éviter les conflits 	<ul style="list-style-type: none"> - Le composant Java boucle tant qu'aucune cible n'est définie - Tous les fichiers sont pris en charge dès leur entrée dans le dossier - Fichier perdu en cas d'arrêt du processus. 	

5.4 N°3 : Betasuisse (JSON)

Nom du flux	mule.rts.betasuisse
Contexte	Lors de la numérisation d'un média par Betasuisse, le prototype fait le lien entre l'identifiant de média situé dans un fichier XML généré pendant la numérisation et le numéro de la cassette physique contenant la vidéo, au travers d'un appel HTTP à une application gérant la base de données (Solr).
Description	<p>Déclencheur : Fichier entrant dans le dossier surveillé.</p> <p>Processus : Génération de fiche XML via appel Solr.</p> <p>Résultat : Nouveau fichier XML.</p> <p>Lorsqu'un fichier entre dans le dossier surveillé, il est convertit au format String et le numéro du média est extrait à l'aide d'une instruction XPATH. Une requête au format statique est générée sur la base de ce numéro de média et envoyée à Solr. Celui-ci retourne un JSON représentant une émission et contenant plusieurs fichiers. Le payload est convertit au format Object afin d'être manipulé par une classe Java, qui va extraire les informations voulues et générer un fichier XML en sortie.</p>
Contraintes	<p>Le fichier ne doit pas être perdu en cas d'arrêt du processus.</p> <p>Traitements simultanés : 1 (processus synchrone).</p> <p>Solr retourne une représentation JSON.</p>
Préconditions	<p>Le fichier XML est considéré valide.</p> <p>La base de données Solr est accessible.</p>

Modèles

Figure 20 : Modèle UML Betasuisse JSON

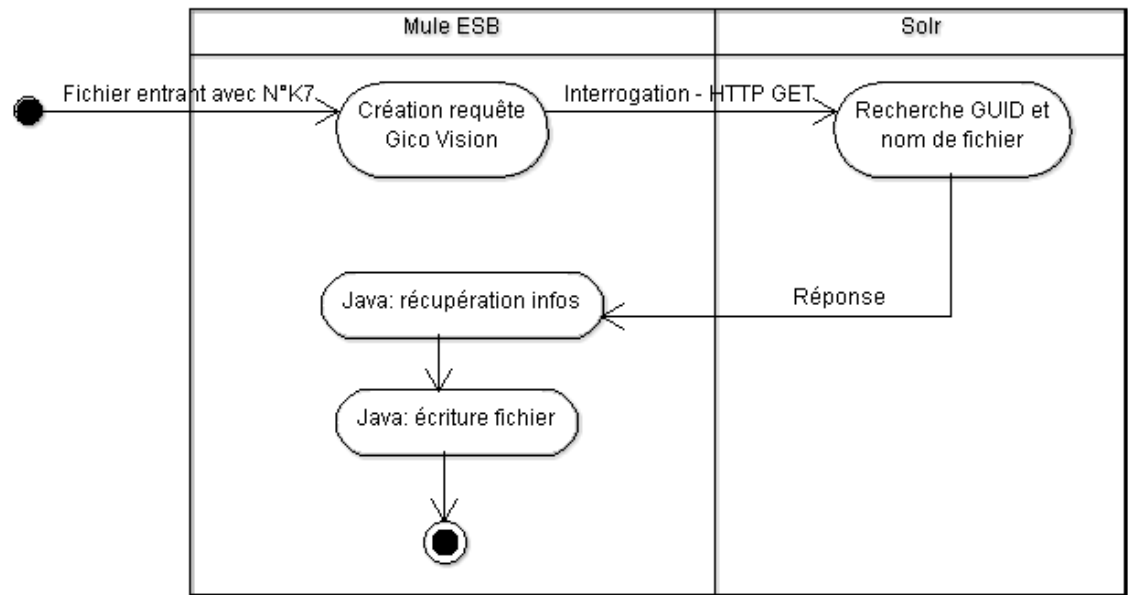
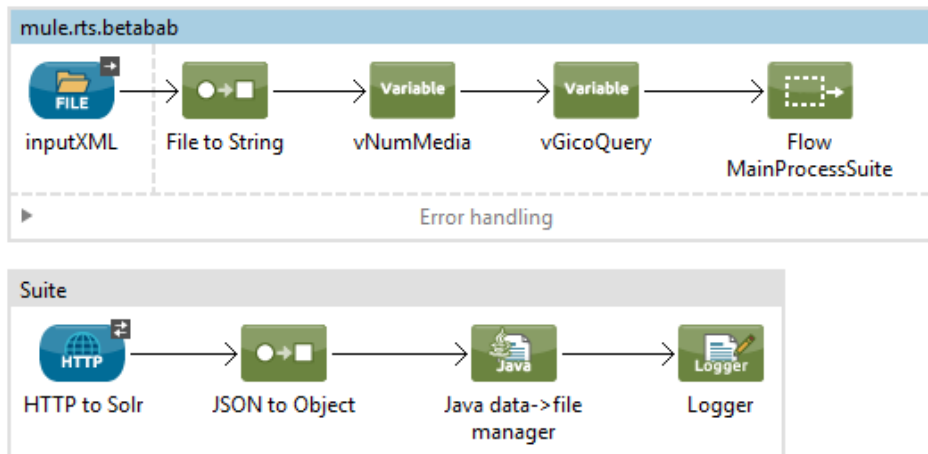


Figure 21 : Modèle Mule Betasuisse JSON



Entrée	/metasem/betasuisse/telechargement/in
	<pre> <?xml version="1.0" encoding="iso-8859-1" standalone="yes"?> <Metadata_File> <identification> <nameUE>RTS</nameUE> <losNr>Lot009Caisse236</losNr> <migrationBegin>5.12.2014</migrationBegin> </identification> <Media_Identification> <fileStatus>Correct</fileStatus> <sMediaID>LM112872_02</sMediaID> <mediaFormat>BETA-SP</mediaFormat> <tMediaID>BS0676L4</tMediaID> </Media_Identification> <BIV> <item> <itemNr>1</itemNr> <startOTC>12:44:43:00</startOTC> <endOTC>12:48:46:00</endOTC> </item> </BIV> </Metadata_File> </pre>
Sortie	/metasem/betasuisse/telechargement/out
<p>Solr :</p> <pre> { "responseHeader":{ "status":0, "QTime":3, "params":{ "fl":"idSupport,UmidEtPosition,UmidEtPositionMediaOriginal", "indent":"true", "q":"(!q.op=AND)UmidEtPositionMediaOriginal:LM112872/02", "wt":"json"}}, "response":{"numFound":1,"start":0,"docs":[{ "UmidEtPositionMediaOriginal":["LM112872/02", "vide", "vide"], "idSupport":["ZB129437", "XV002400", "LM112872"], "UmidEtPosition":["ZB129437/01", "XV002400/02", "LM112872/02"]}] }} </pre> <p>Java Component :</p>	

```

<?xml version="1.0" encoding="UTF-8"?>
<stock>
  <edob ref="1" externalIdType="GUID" externalId="GE1003333870" />
  <supportFile ref="2" externalIdType="UMID" externalId="ZB129437">
    <SupportFileOriginalFilename>LM112872_02.mxf</SupportFileOriginalFilename>
  </supportFile>
  <mediaVideo ref="3" externalIdType="UMIDPOS" externalId="ZB129437/01" />
</stock>

```

Composant	Nom	Rôle
Connector HTTP	HTTP->Solr	Envoie la requête créée dynamiquement dans les variables au serveur Solr et récupère la réponse.
Transformers JsonObject	JsonObject	Convertit un objet org.mule.transport.file.FileMessageReceiver en java.util.LinkedHashMap.
Transformers Variable	vNumMedia	Nom original du support physique cherché.
	vGicoQuery	Requête adressée au serveur Solr.
Component Java	Java data -> File manager	Appelle la classe BabAnalyzer dont l'algorithme cherche et extrait les données de la LinkedHashMap afin de générer un fichier XML.

Analyse du prototype

Ce prototype a permis de mettre en avant le cycle de vie du *Payload* tout au long du flux. En effet, celui-ci va se transformer de fichier à résultat HTTP lors du passage dans ce connecteur. Il n'est donc plus possible de faire communiquer le composant Java et le connecteur *File* de sortie. De plus la structure renvoyée n'est pas un JSON exploitable directement et ne propose donc pas une réelle amélioration de performances. Autre facteur clé : le transformateur *JSON To XML* ne fournit que la moitié de la réponse renvoyée par le connecteur HTTP, rendant ce transformateur inutilisable dans ce cas. Le traitement du JSON n'est donc pas une fonctionnalité très aboutie dans Mule 3.6.

<ul style="list-style-type: none">+ Utilisation du composant HTTP et traitement du retour+ Requête dynamique basée sur une instruction XPATH+ Double utilisation de connecteurs	<ul style="list-style-type: none">- Méthode Java qui écrit le fichier résultat- JSON To XML n'est pas utilisable- Difficulté de maintenance- Structure de données retour HTTP- Mauvaise gestion des erreurs
---	---

5.5 N°4 : Betasuisse (XML)

Nom du flux	mule.rts.betasuisse-xml
Contexte	Fait le lien entre un identifiant de média situé dans un fichier XML et le numéro de la cassette physique contenant la vidéo, au travers d'un appel HTTP à une application gérant la base de données (Solr).
Description	<p>Déclencheur : Fichier entrant dans le dossier surveillé.</p> <p>Processus : Génération de fiche XML via appel Solr.</p> <p>Résultat : Nouveau fichier XML.</p> <p>Lorsqu'un fichier entre dans le dossier surveillé, il est convertit au format String et le numéro du média est extrait à l'aide d'une instruction XPATH. Une requête au format statique est générée sur la base de ce numéro de média et envoyée à Solr. Celui-ci retourne un XML (Stream par le composant HTTP) représentant une émission et contenant plusieurs fichiers. Le payload est convertit au format XML afin d'être manipulé par deux transformateurs XSLT, qui vont extraire les informations voulues et générer un fichier XML en sortie.</p>
Contraintes	<p>Le fichier ne doit pas être perdu en cas d'arrêt du processus.</p> <p>Traitements simultanés : 1 (processus synchrone).</p> <p>Solr retourne une représentation XML.</p>
Préconditions	<p>Le fichier XML est considéré valide.</p> <p>La base de données Solr est accessible.</p>

Modèles

Figure 22 : Modèle UML Betasuisse XML

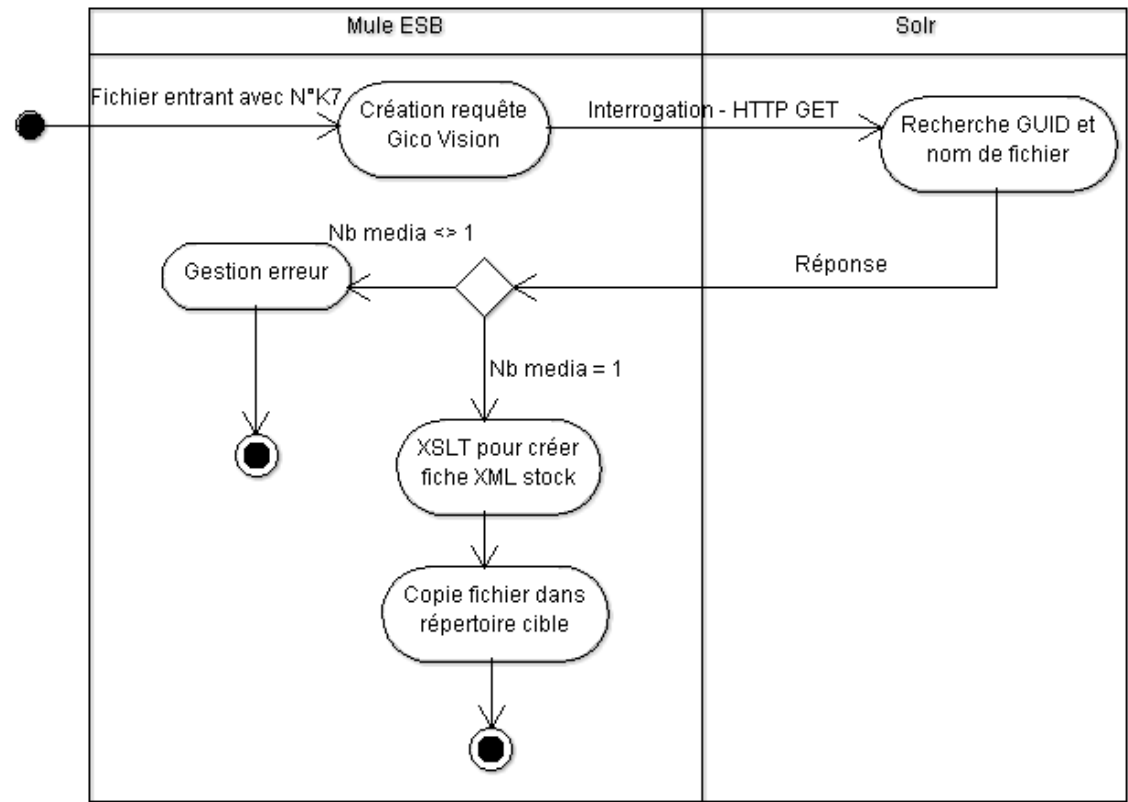
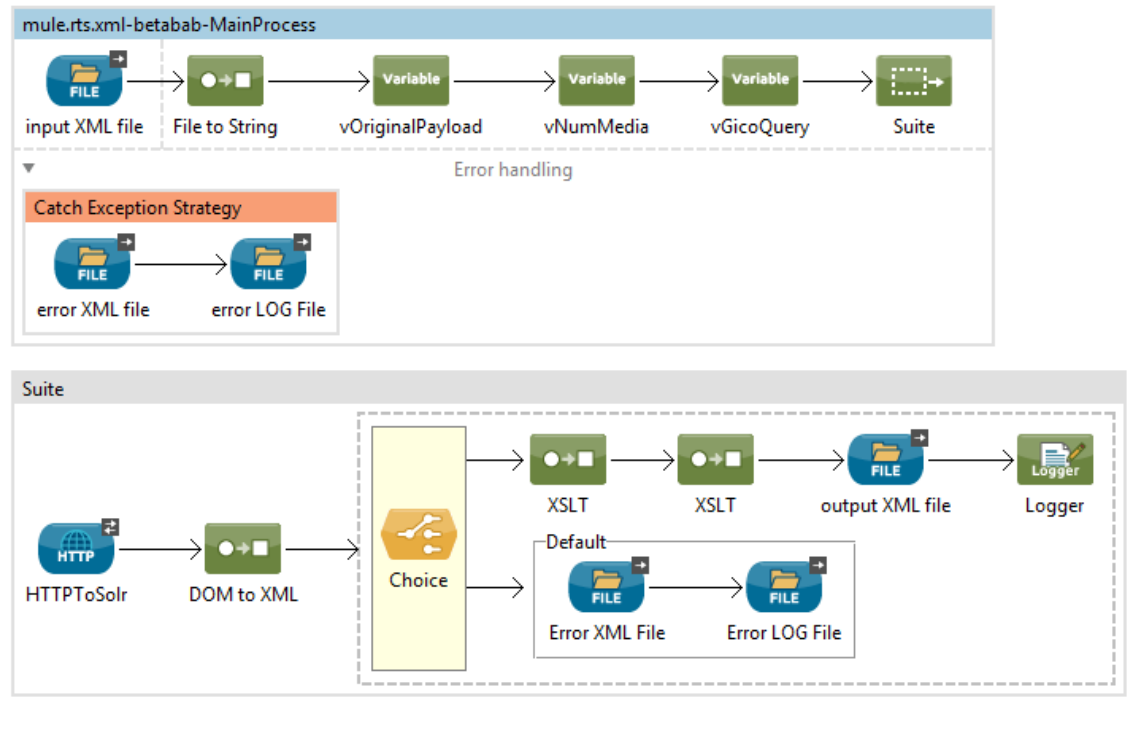


Figure 23 : Modèle Mule Betasuisse XML



Entrée

/metasem/betasuisse/telechargement/in

```
<?xml version="1.0" encoding="iso-8859-1" standalone="yes"?>
<Metadata_File>
  <identification>
    <nameUE>RTS</nameUE>
    <losNr>Lot009Caisse236</losNr>
    <migrationBegin>5.12.2014</migrationBegin>
  </identification>
  <Media_Identification>
    <fileStatus>Correct</fileStatus>
    <sMediaID>LM112872_02</sMediaID>
    <mediaFormat>BETA-SP</mediaFormat>
    <tMediaID>BS0676L4</tMediaID>
  </Media_Identification>
  <BIV>
    <item>
      <itemNr>1</itemNr>
      <startOTC>12:44:43:00</startOTC>
      <endOTC>12:48:46:00</endOTC>
    </item>
  </BIV>
</Metadata_File>
```

Sortie

/metasem/betasuisse/telechargement/out

Solr :

```
<response>
  <lst name="responseHeader">
    <int name="status">0</int>
    <int name="QTime">0</int>
    <lst name="params">
      <str name="fl">idSupport,UmidEtPosition,UmidEtPositionMediaOriginal</str>
      <str name="indent">>true</str><str name="q">(!q.op=AND)UmidEtPositionMediaO
      <str name="wt">xml</str>
    </lst>
  </lst>
  <result name="response" numFound="1" start="0">
    <doc>
      <arr name="UmidEtPositionMediaOriginal">
        <str>LM112872/02</str>
        <str>vide</str>
        <str>vide</str>
      </arr>
      <arr name="idSupport">
        <str>ZB129437</str>
        <str>XV002400</str>
        <str>LM112872</str>
      </arr>
      <arr name="UmidEtPosition">
        <str>ZB129437/01</str>
        <str>XV002400/02</str>
        <str>LM112872/02</str>
      </arr>
    </doc>
  </result>
</response>
```

XSLT :

```

<?xml version="1.0" encoding="UTF-8"?>
<stock>
  <edob ref="1" externalIdType="GUID" externalId="GE1003333870" />
  <supportFile ref="2" externalIdType="UMID" externalId="ZB129437">
    <SupportFileOriginalFilename>LM112872_02.mxf</SupportFileOriginalFilename>
  </supportFile>
  <mediaVideo ref="3" externalIdType="UMIDPOS" externalId="ZB129437/01" />
</stock>

```

Composant	Nom	Rôle
Connector HTTP	HTTP->Solr	Envoie la requête au serveur Solr et récupère la réponse.
Transformers Variable	vNumMedia	Nom original du support physique cherché.
	vGicoQuery	Requête adressée au serveur Solr.
Transformers XSLT	XSLTIndex	Génère pour chaque <str> de <doc> un index.
	XSLTChange	Génère un nouveau fichier.

Analyse du prototype

Ce prototype a permis de démontrer que l'utilisation de fiches XML et de transformateurs XSLT est bien plus efficacement prise en charge dans Mule que le format JSON. La différence est de l'ordre de la moitié du temps pour la réalisation de ce prototype en comparaison de son équivalent JSON. Le bon format de données est donc essentiel pour une réalisation rapide, cohérente et efficace.

- + Optimisation du prototype Betasuisse JSON
- + Le traitement *XML/XSLT* est performant et efficace.
- + Prise en charge/écriture du fichier par le composant *File*

– Aucun aspect négatif relevé

5.6 N°5 : Move and Rename

Nom du flux	mule.rts.move-and-rename
Contexte	Ce prototype permet de gérer différentes applications avec un seul dossier pour point d'entrée. On peut ainsi router une fiche XML sur la base de son chemin et/ou de son nom de fichier. C'est utilisé par exemple dans le secteur de l'Actualité.
Description	<p>Déclencheur : Fichier entrant dans le dossier surveillé.</p> <p>Processus : Renommage et déplacement de fiche.</p> <p>Résultat : Fichier renommé et déplacé.</p> <p>L'application doit permettre, lors de l'entrée d'une fiche XML dans l'un des dossiers surveillé, de récupérer le nom de l'application dont provient cette fiche. Le fichier est renommé puis déposé dans le dossier d'input de l'actualité. Pour ajouter un nouveau dossier : modifier les fichiers de propriétés et ajouter un nouveau flux qui contient un connecteur de type <i>File</i> dont le point d'entrée sera le nouveau dossier voulu, et un <i>Flow Reference</i> vers l'ajout de préfixe. L'application se charge du reste. Règles de renommage :</p> <p>Aller :</p> <ul style="list-style-type: none"> • Pour : /data/{environnement}/metasem/application/type/in/fichier.xml • Renommage : #application#type#fichier.xml • Dépôt : /data/{environnement}/metasem/sonaps/import_news/in/ <p>Retour :</p> <ul style="list-style-type: none"> • Pour : /data/{environnement}/metasem/sonaps/import_news/[ok ko]/ #application#type#fichier.xml • Renommage : fichier.xml • Dépôt : /data/{environnement}/metasem/application/type/[ok ko]/
Contraintes	<p>Le fichier ne doit pas être perdu en cas d'arrêt du processus.</p> <p>Traitements simultanés : 1 (processus synchrone).</p>
Préconditions	Le fichier XML est considéré comme valide.

Modèles

Figure 24 : Modèle UML Move and Rename

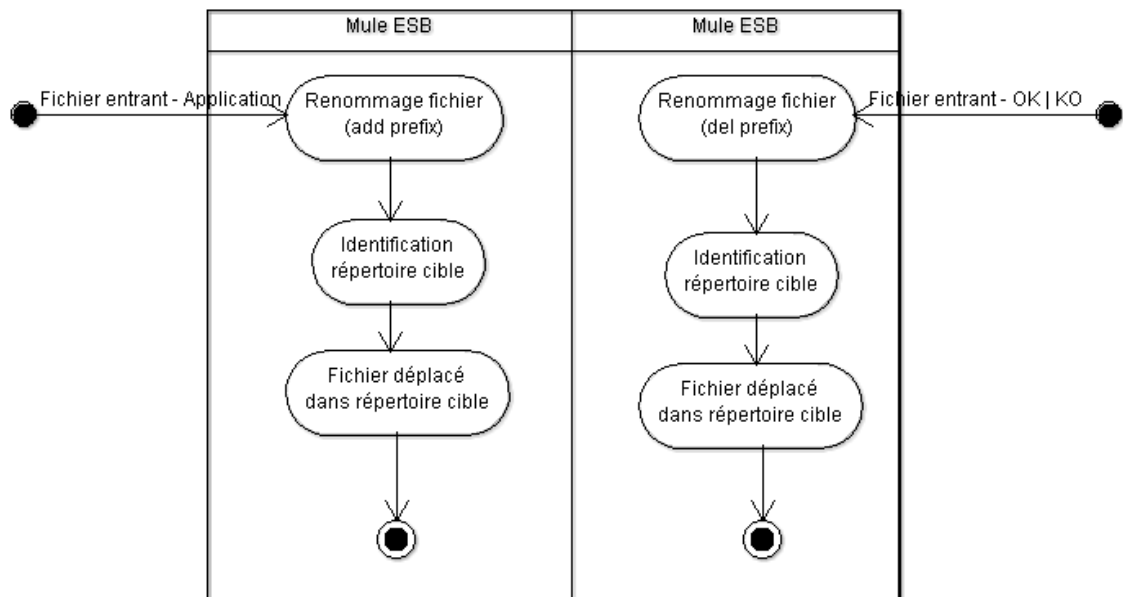
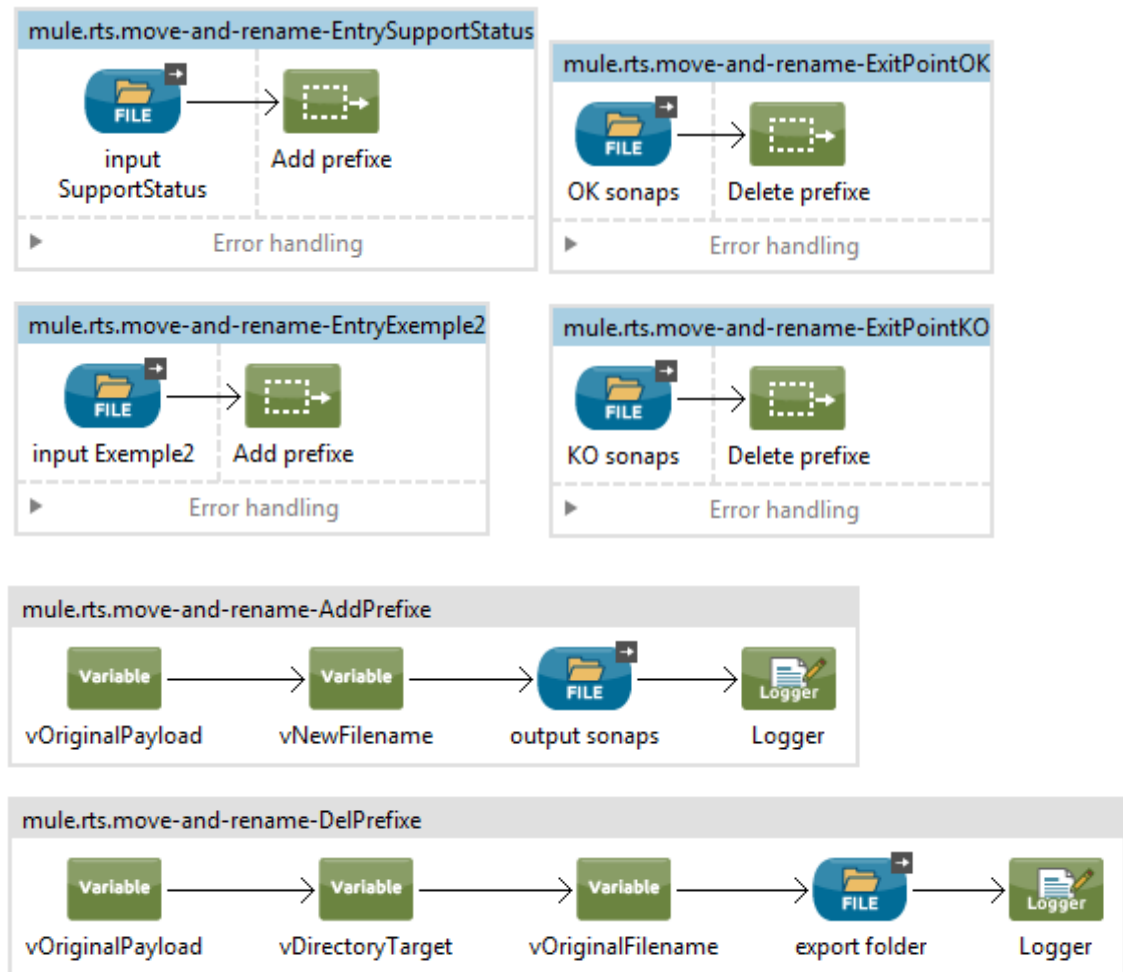
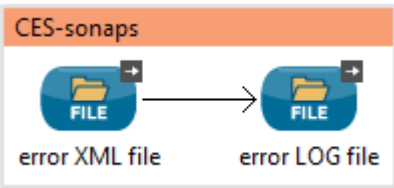


Figure 25 : Modèle Mule Move and Rename



		
Entrée /metasem/{application/type}/in		
Fichier entrant.		
Sortie /metasem/{application/type}/[ok ko]		
Fichier renommé et déposé au bon endroit selon la règle de renommage décrite plus haut.		
Composant	Nom	Rôle
Scope Subflow	AddPrefixe	Contient les composants qui ajoutent le préfixe.
	DelPrefixe	Contient les composants qui enlèvent le préfixe.
Scope Flow	EntryApp (1 par dossier surveillé)	Contient le dossier surveillé et une référence au subflow <i>AddPrefixe</i> . C'est ce type de flow que l'on va dupliquer pour ajouter un nouveau dossier surveillé.
	ExitPoint	Contient le dossier surveillé et une référence au subflow <i>DelPrefixe</i> .
Connector File	input-App	Surveille <u>un</u> dossier d'input.
		Ecrit les fichiers dans le dossier de backup
	output Sonaps	Ecrit le fichier préfixé dans le dossier d'input de sonaps
	export folder	Ecrit le fichier non préfixé dans le dossier de résultat de l'application de base.

Transformers Variable	vNewFilename	Contient le nom préfixé.
	vDirectoryTarg et	Contient le répertoire d'origine du fichier.
	vOriginalFilena me	Contient le nom d'origine du fichier
Analyse du prototype		
<p>Ce prototype a permis de mettre en avant le fait qu'un point d'entrée ne peut pas être dynamique. Le composant <i>File</i>, pour être un point d'entrée et non de sortie doit être le premier composant du flux. On ne peut donc pas assigner la valeur de son point d'entrée dynamiquement au travers de variables ou autres comme pour les points de sortie. Afin que l'application fonctionne, on est obligé de disposer d'un point d'entrée par dossier surveillé, et d'ajouter celui-ci dans les fichiers de propriétés.</p>		
<ul style="list-style-type: none"> + Traitement qui permet de n'avoir qu'un point d'arrivée pour de multiples applications. + Traitements sur les noms de fichier 		<ul style="list-style-type: none"> - Difficulté d'ajouter une application surveillée - 1 point d'entrée / application (input <i>File</i> non dynamique)

5.7 N°6 : Import Récursif

Nom du flux	mule.rts.import-recursif
Contexte	<p>Ce prototype se base sur le prototype « Move and Rename ». Le but est de faire la même manipulation de renommage, mais de façon récursive. Pour faire ceci, une modification de l'arborescence est nécessaire. En effet, la méthode récursive prendrait chaque fichier présent dans import-in, on ne peut donc pas mettre les fichiers de backup et de résultat dans l'un de ses sous-dossiers.</p> <p><u>Actuellement :</u></p> <p style="text-align: center;">/metasem/application/type/ [in ok ko history] /</p> <p><u>Nécessaire :</u></p> <p style="text-align: center;">/metasem/import-in/application/type/in/</p> <p style="text-align: center;">/metasem/import-result/application/type/ [ok ko history]/</p>
Description	<p>Déclencheur : Fichier entrant dans le dossier surveillé.</p> <p>Processus : Renommage et déplacement de fiche.</p> <p>Résultat : Fichier renommé et déplacé.</p> <p>Le prototype doit réaliser exactement la même action que le prototype « Move and Rename », selon les mêmes règles mais en permettant de n'avoir qu'un seul point d'entrée pour toutes les applications et permettant donc d'ajouter en une seule étape (création des dossiers de base dans l'arborescence) une application à la liste de celles qui sont prise en charge.</p>
Contraintes	<p>Le fichier ne doit pas être perdu en cas d'arrêt du processus.</p> <p>La prise en charge d'un nouveau dossier doit être automatique.</p> <p>Traitements simultanés : 1 (processus synchrone).</p>
Préconditions	<p>L'arborescence a été modifiée.</p> <p>Le fichier est considéré valide.</p>

Modèles

Figure 26 : Modèle UML Import Récuratif

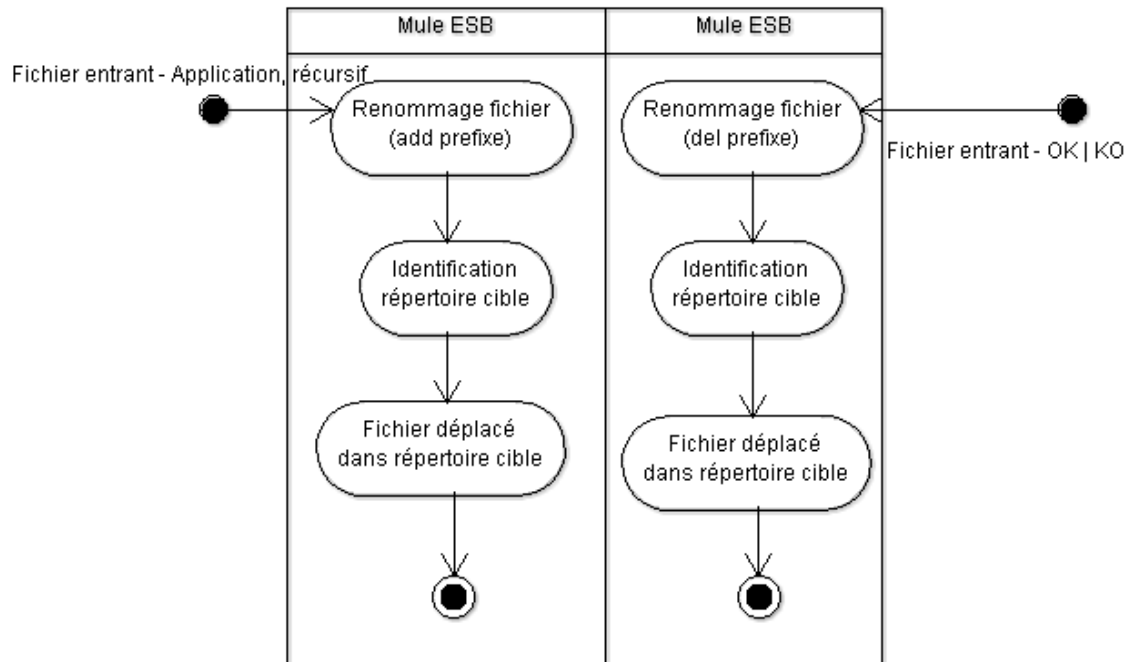
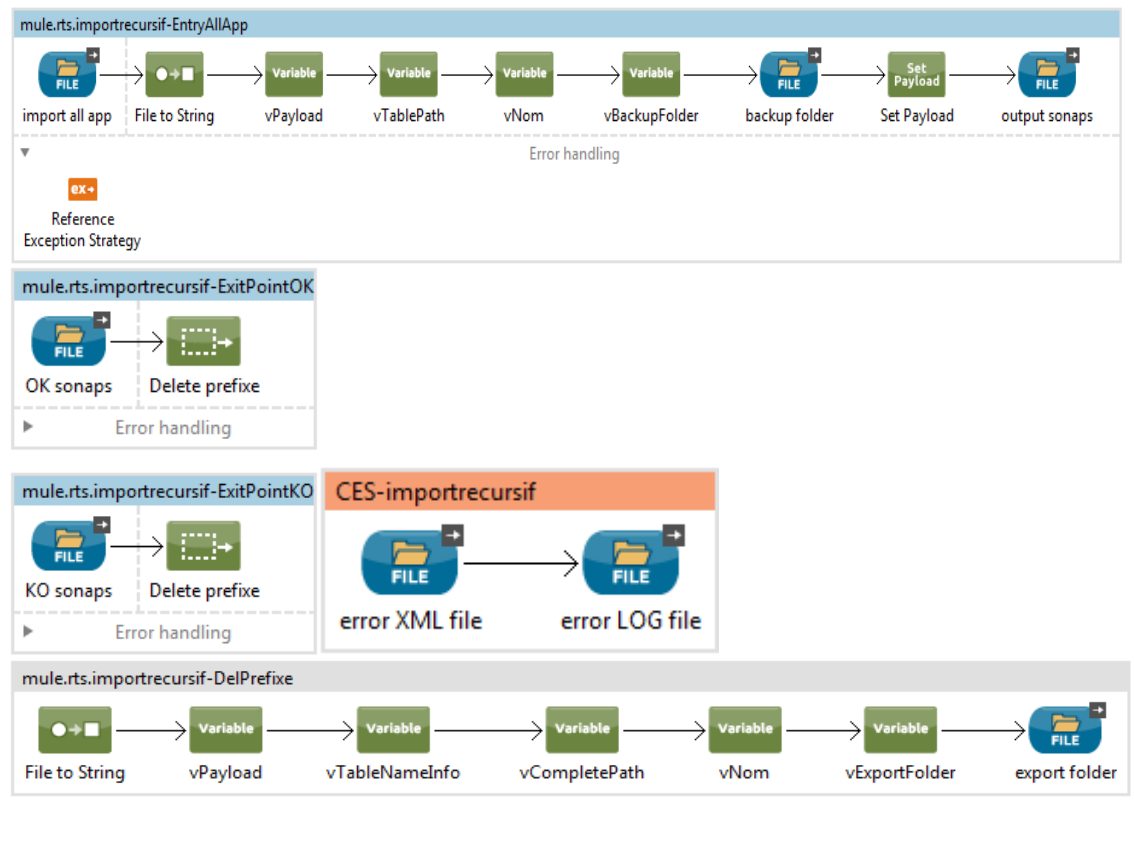


Figure 27 : Modèle Mule Import Récuratif



Entrée /metasem/import-in/{application/type}/in		
Fichier entrant.		
Sortie /metasem/import-result/{application/type}/[ok ko]		
Fichier renommé et déposé au bon endroit selon la règle de renommage décrite plus haut.		
Composant	Nom	Rôle
Scope Subflow	AddPrefixe	Contient les composants qui ajoutent le préfixe.
	DelPrefixe	Contient les composants qui enlèvent le préfixe.
Connector File	import all app	Surveille le dossier d'input de façon récursive.
	output sonaps	Ecrit le fichier préfixé dans le dossier d'input de sonaps.
	[OK KO] sonaps	Récupère les fichiers traités par Sonaps.
	export folder	Ecrit le fichier non préfixé dans le dossier de résultat de l'application de base.
Transformers Variable	vTablePath	Sépare (à chaque « / ») les différents composants du chemin du fichier en un tableau.
	vNom	(Flux <i>AddPrefixe</i>) Génère le nom de fichier sur la base des informations de <i>vTablePath</i> .
		(Flux <i>DelPrefixe</i>) Génère le nom non-préfixé sur la base des informations de <i>vTableNameInfo</i> .
	vBackupFolder	Génère le chemin du dossier de backup sur la base des informations de <i>vTablePath</i>

	vTableNameInfo	Sépare (à chaque « # ») les différents composants du nom de fichier en un tableau.
	vCompletePath	Sépare (à chaque « / ») les différents composants du chemin du fichier en un tableau.
	vExportFolder	Génère le chemin du dossier d'output de l'application originale.

Analyse du prototype

Ce prototype a permis de mettre en avant les contraintes du composant *File* et particulièrement de son dossier de backup. En effet, comme l'application ne sait pas d'avance la profondeur de l'arborescence et sa position à l'intérieur de celle-ci, il est impossible de définir un dossier de backup dynamique au point d'entrée. Il faut donc enregistrer ce fichier plus tard, quand ces informations ont pu être récoltées.

- | | |
|--|---|
| <ul style="list-style-type: none"> + Optimisation du prototype « Move-and-Rename ». + Traitement Java dans les variables, aucun composant ni class Java. + Facilité d'ajout d'une application (souplesse) | <ul style="list-style-type: none"> - Modification de l'arborescence nécessaire |
|--|---|

5.8 N°7 : Thumbfile Requester

Nom du flux	mule.rts.thumbfile-requester
Contexte	Ce prototype se base sur le prototype « Thumbfile Dispatch » et propose une amélioration de son système de gestion des fichiers. Le but de cette amélioration est d'activer le composant d'input uniquement si de la place est disponible dans l'un des dossiers cible, et de ne faire tourner le processus que si toutes les conditions sont réunies, contrairement au prototype précédent qui bouclait jusqu'à trouver un dossier de destination.
Description	<p>Déclencheur : Timer (20 secondes).</p> <p>Processus : Répartition de charge.</p> <p>Résultat : Fichier déplacé, charge répartie.</p> <p>Toutes les 20 secondes, le composant <i>Poll</i> lance l'application dans le but de déposer un fichier tout en respectant la charge maximum de ceux-ci. Pour effectuer cette tâche, on va tout d'abord chercher le nombre de fichiers en attente de traitement avec la méthode <i>getNbFichiers</i> de la classe <i>DispatcherTools</i>. S'il y a au moins un fichier, on va chercher le dossier de destination via la méthode <i>getDossierCible</i> de la même classe. Si un dossier a été défini (ce qui assure que la place est disponible), on charge le fichier depuis le module complémentaire <i>Mule Requester</i> téléchargé en ligne. Ce composant permet de récupérer un fichier à n'importe quel moment du flux, à l'inverse du connecteur <i>File</i> qui ne permet cette action que s'il est le premier élément (la source) du flux et n'autorise donc pas les vérifications en amont. Le fichier est ensuite copié dans son dossier de destination défini dynamiquement.</p>
Contraintes	<p>Le fichier ne doit pas être perdu en cas d'arrêt du processus.</p> <p>Le fichier ne doit pas être pris en charge si aucune place n'est disponible.</p>

	<p>Le processus doit être déclenché par un timer (<i>poll</i>).</p> <p>Traitements simultanés : 1 (processus synchrone)</p> <p>Maximum de fichiers par dossier : 5</p> <p>Nombre de dossiers possibles : 3</p> <p>Nombre d'essais possibles : illimités</p>
--	---

Préconditions	Le fichier reçu est considéré valide.
----------------------	---------------------------------------

Modèles

Figure 28 : Modèle UML Thumbfile Requester

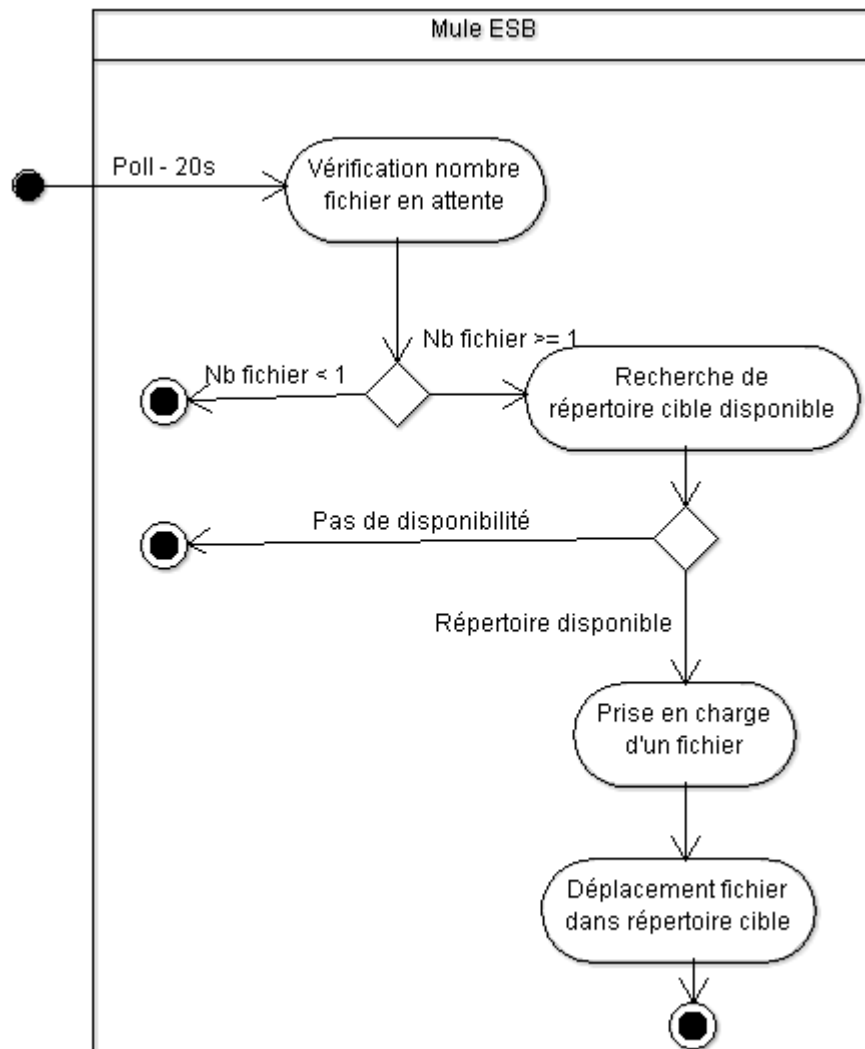
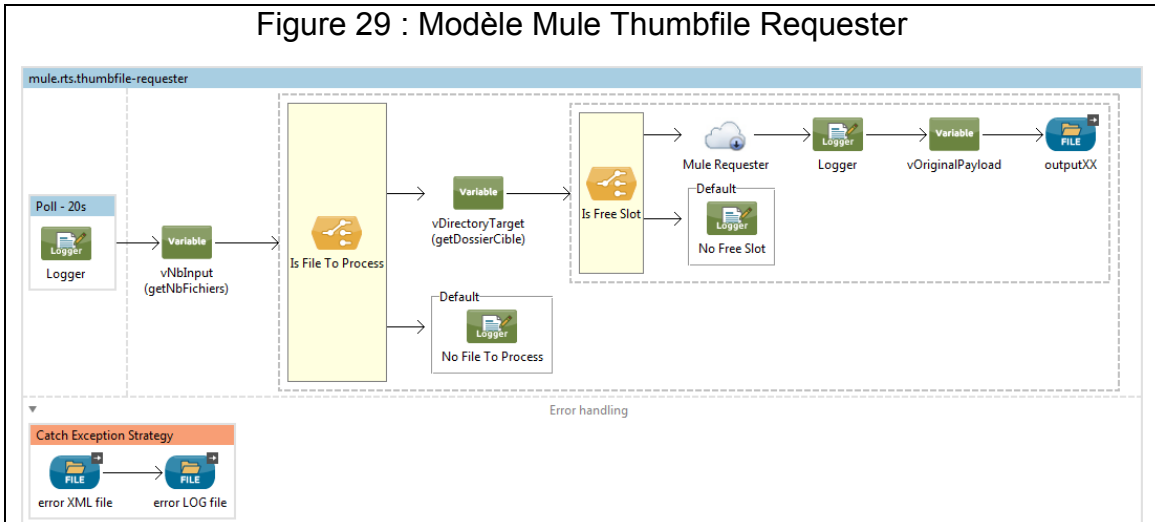


Figure 29 : Modèle Mule Thumbfile Requester



Entrée /metasem/dam/thumb/in

Fichier XML dans le dossier de temporisation.

Sortie /metasem/dam/thumb/[in01 | in02 | in03]

Fichier XML copié dans le dossier d'output défini.

Composant	Nom	Rôle
Transformers Variable	vNbInput	Appelle une fonction Java chargée de vérifier qu'il y a des fichiers en attente de traitement
	vDirectoryTarget	Appelle une fonction Java chargée de définir le dossier cible (le moins chargé)
Flow Control Choice	Is File To Process	Laisse passer le message si des fichiers sont en attente de traitement. Sinon, affiche un message.
	Is Free Slot	Laisse passer le message si une cible a pu être définie. Sinon, affiche un message.
Module complémentaire	Mule Requester	Prend un objet en source en cours de flux, ici un fichier.

Analyse du prototype

La première difficulté provient du fait qu'initialement, dans le précédent flux gérant les vignettes, c'est lorsqu'un fichier entre (et est donc pris en charge) que se déclenche l'évènement qui va déterminer le nombre de fichiers par dossier. Le fichier est donc déjà pris et ne plus être relâché, ce qui entraîne sa destruction en cas d'erreur, d'annulation ou autre évènement qui empêcherai le processus de se terminer correctement.

La deuxième difficulté, c'est le comportement du composant *File*, qui charge tous les fichiers présents dans le dossier en mémoire lors de son activation et nécessite d'être placé en première position. Ce problème est contourné en utilisant le *Mule Requester* qui ne prend qu'un fichier à la fois, et à n'importe quel moment du flux.

- + Optimisation du prototype « Thumbfile Dispatch »
- + Traitement effectué que si les conditions le permettent
- + Fichier conservé même en cas d'erreur ou d'arrêt du prototype
- + Traitement basé sur un timer, indépendant de l'entrée de fichiers dans le dossier
- + Pas de traitement en boucle
- + Extension des possibilités de Mule avec installation d'un nouveau composant

- S'il n'y a pas de place ou pas de fichier, attente de 20s avant de réessayer.
- Mauvaise maîtrise du composant *Mule Requester*

5.9 N°8 : Sparkex

Nom du flux	mule.rts.sparkex
Contexte	Ce flux assure la synchronisation entre deux systèmes de fichier (le DAM et le SPARKEX). Aucune transformation n'est effectuée dans ce flux, le principal intérêt de cette application étant de tester la résistance de Mule face à la charge. En effet, ce processus devrait traiter environ 20'000 à 30'000 fiches par jour, ce qui représente une charge bien plus élevée que dans les autres prototypes.
Description	Déclencheur : Fichier entrant dans le dossier surveillé. Processus : Déplacement de nombreuses fiches, test de charge. Résultat : Fichier déplacé.
	Chaque fichier entrant dans l'un des 4 dossiers surveillés est simplement copié dans le dossier statique qui lui correspond sur le système de fichier parallèle.
Contraintes	Le fichier ne doit pas être perdu en cas d'arrêt du processus.
Préconditions	Les deux systèmes de fichiers sont opérationnels et accessibles.

Modèles

Figure 30 : Modèle UML Sparkex

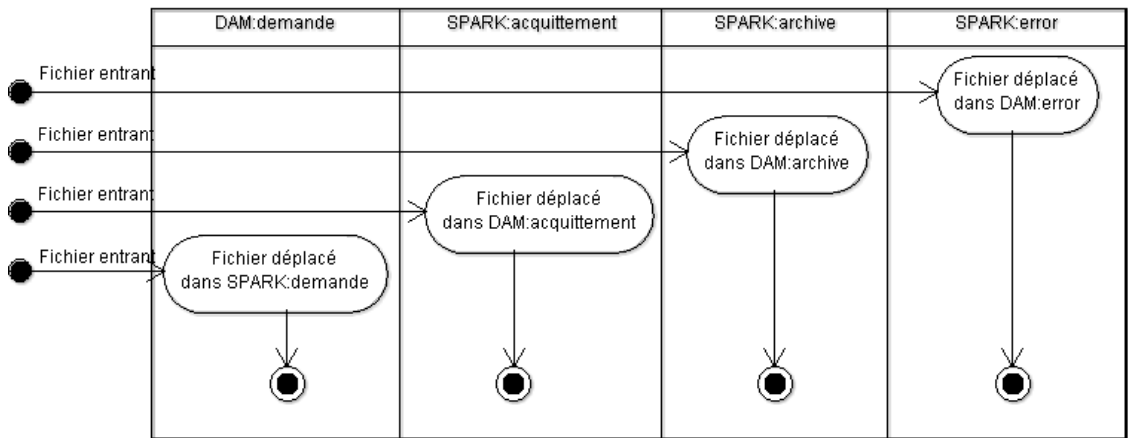
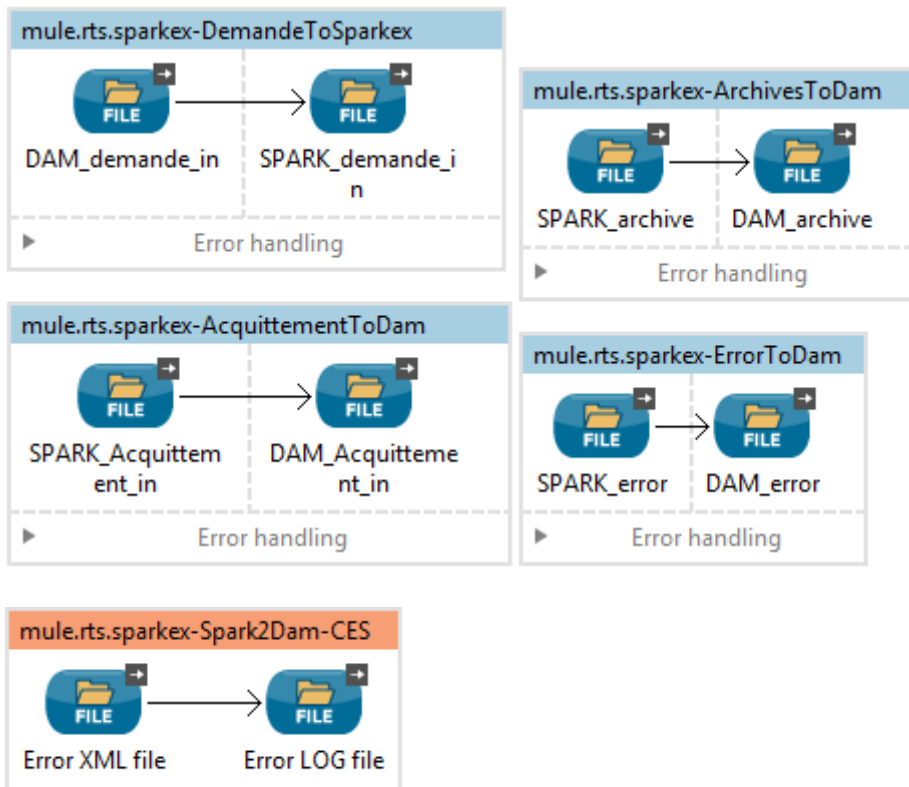


Figure 31 : Modèle Mule Sparkex



Entrée (SPARKE X) /metasem/dam/[demande | acquittement]/in

Fichier XML

Sortie (DAM) /dam/[demande | acquittement]/in

Fichier XML déplacé		
Composant	Nom	Rôle
Connecteur File	-	Tous les connecteurs assurent la synchronisation, aucun traitement n'est effectué.
Analyse du prototype		
Ce prototype extrêmement simple permet d'évaluer la capacité de Mule à gérer des quantités moyennes de fiches.		
+ Test de charge		- Aucun traitement

5.10 N°9 : MassDL

Nom du flux	mule.rts.massdl
Contexte	Ce processus gère le lancement et la charge du traitement des téléchargements et archivages en tâche de fond. Le but de l'application est de permettre une gestion de la charge et un téléchargement continu des médias en attente. Si les ressources le permettent, l'application va lancer un téléchargement. L'exécution se fait lorsque le système n'est ni surchargé en téléchargements ni en archivage.
Description	<p>Déclencheur : Timer (20 secondes)</p> <p>Processus : Lancement de téléchargement en tâche de fond.</p> <p>Résultat : Téléchargements dans Gico.</p> <p>Toutes les 20 secondes, le composant <i>Poll</i> va lancer la vérification du système qui consiste à vérifier que l'on peut lancer un téléchargement de fond. Cette vérification se base sur plusieurs critères : premièrement, la capacité totale ne doit pas être atteinte en téléchargement et en archivage quelle que soit la tâche envisagée. Deuxièmement, afin de ne pas saturer le système, chaque fichier référence un nom de template qui est lui aussi limité en quantité simultanée. On vérifie donc que cette limite ne soit pas atteinte lors de la lecture du fichier. Enfin, on vérifie que le serveur FTP est aussi disponible et non surchargé. Les deux premières vérifications s'effectuent via une requête en base de données pour laquelle on utilise le connecteur <i>y</i> relatif. La vérification FTP consiste à compter le nombre de fichiers présents sur le serveur de destination.</p> <p>Si toutes les vérifications ne se sont pas effectuées correctement, on affiche un message dans la console. Sinon, on passe à l'étape suivante qui consiste à compter le nombre de fichiers en attente de traitement. Si aucun fichier n'est en attente, on affiche un message dans la console. Sinon, on passe à l'étape suivante dans laquelle on charge le fichier. A ce stade, il faut vérifier l'intégrité et la</p>

	<p>cohérence du fichier. Pour ceci, on vérifie qu'aucune balise n'a été laissée vide à l'aide d'un fichier XSD, que tous les champs obligatoires ont été saisis et que l'identifiant du <i>Template</i> correspond bien à celui auquel on s'attend et qu'il n'y a pas trop de ce <i>Template</i> en téléchargements actifs. Enfin, on vérifie que lors d'une restitution partielle les délimiteurs d'entrée et de sortie (<i>TcIn / TcOut</i>) ont bien été définis.</p> <p>Si aucune erreur n'a été révélée, la création de la requête peut commencer. Avant cela, le fichier XML d'origine est enregistré dans une variable afin de pouvoir le transformer à nouveau lors des prochaines étapes. On applique ensuite une transformation XSLT qui génère le payload au format attendu par le Web Service (WS) de création de requête dans Gico. Si la requête s'effectue normalement, on dispose à présent d'un REQ-ID, c'est-à-dire de l'identifiant de la requête qui vient d'être créée. Si on dispose de cet ID, on va ajouter les médias à la requête au travers d'un deuxième appel WS effectué après une nouvelle transformation XSLT qui utilise la REQ-ID. Enfin, lorsque les médias ont été ajoutés, une troisième transformation et un nouvel appel WS est effectué afin de démarrer le téléchargement. Toutes ces actions sont visibles depuis Gico, au travers d'un navigateur web.</p>
<p>Contraintes</p>	<p>Le fichier ne doit pas être perdu en cas d'arrêt du processus.</p> <p>Le fichier ne doit pas être pris en charge si aucune place n'est disponible.</p> <p>Le processus doit être déclenché par un timer (<i>poll</i>).</p> <p>La cohérence du fichier doit être vérifiée.</p> <p>L'état du système doit être vérifié.</p> <p>Traitements simultanés : 1 (processus synchrone)</p>
<p>Préconditions</p>	<p>La base de données est accessible</p> <p>Les Web Services sont accessibles</p>

Figure 32 : Modèle UML MassDL

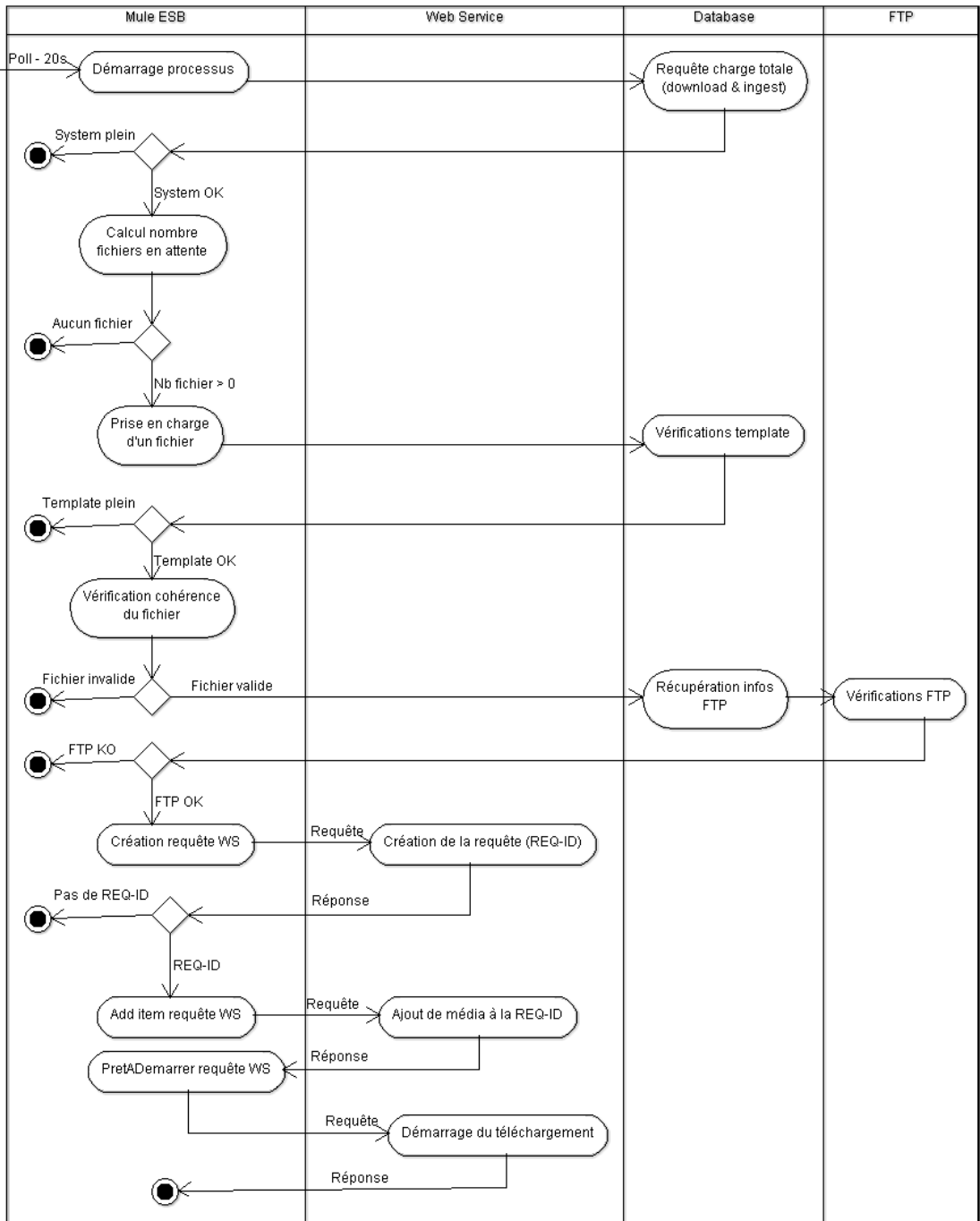
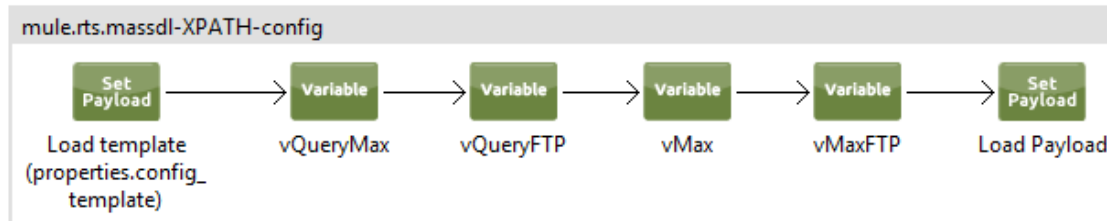
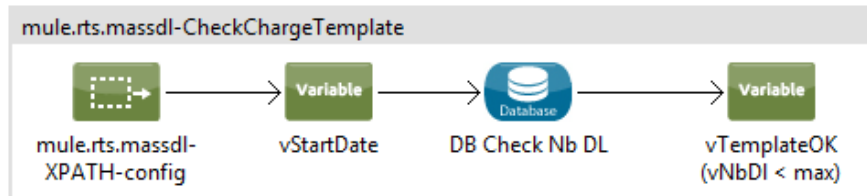
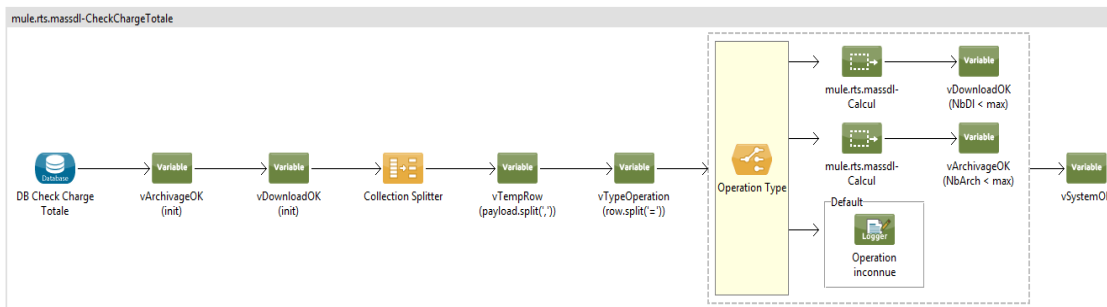
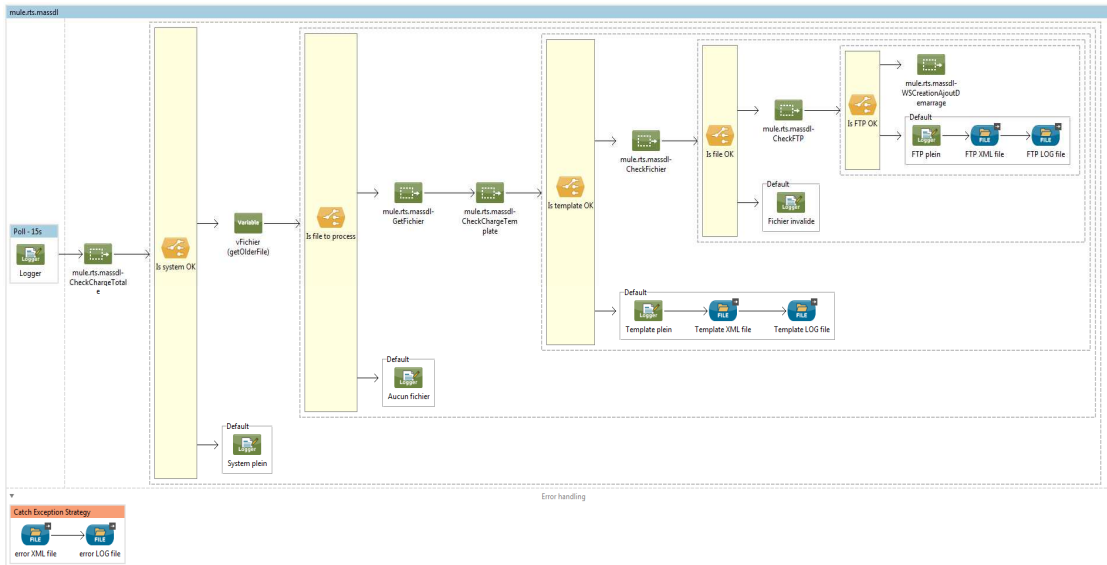
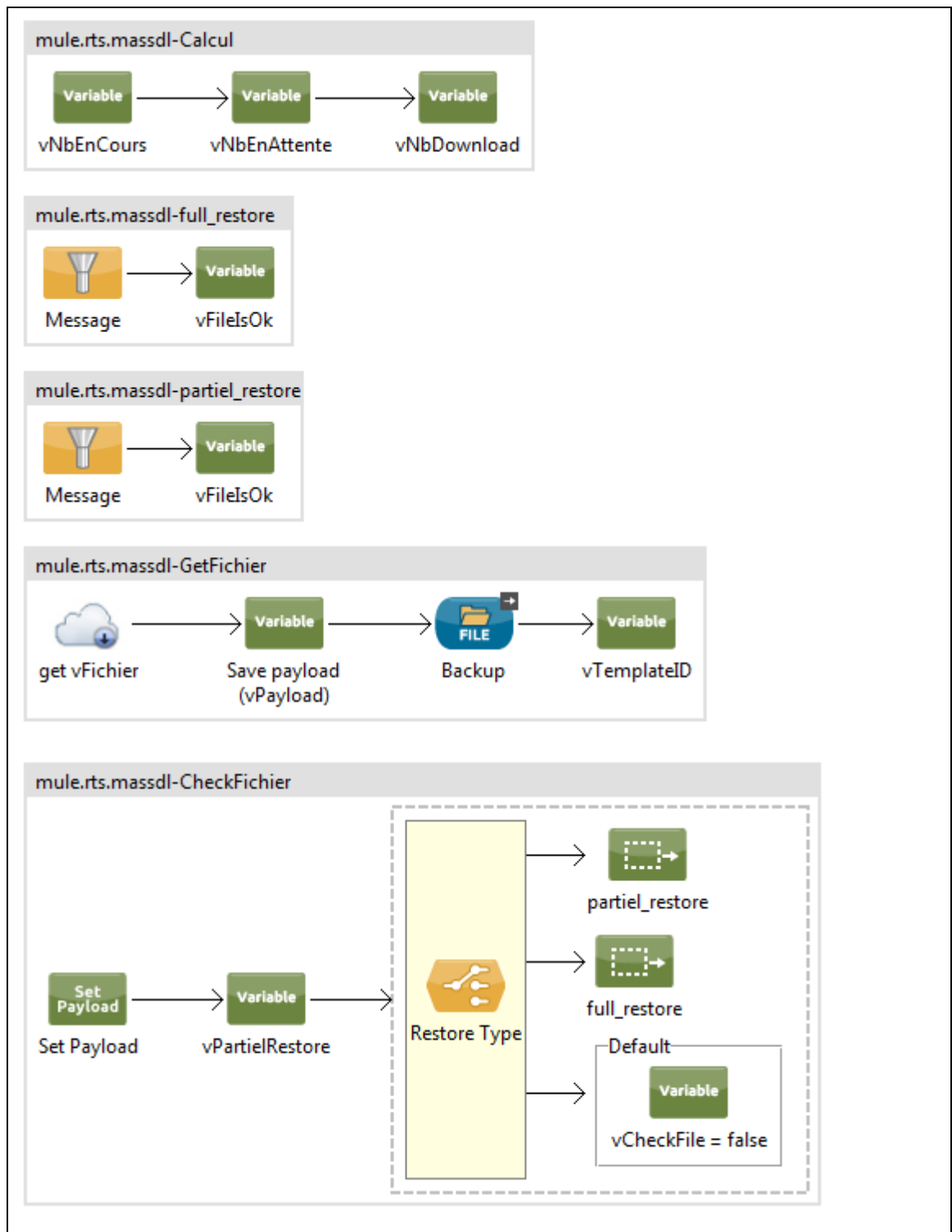
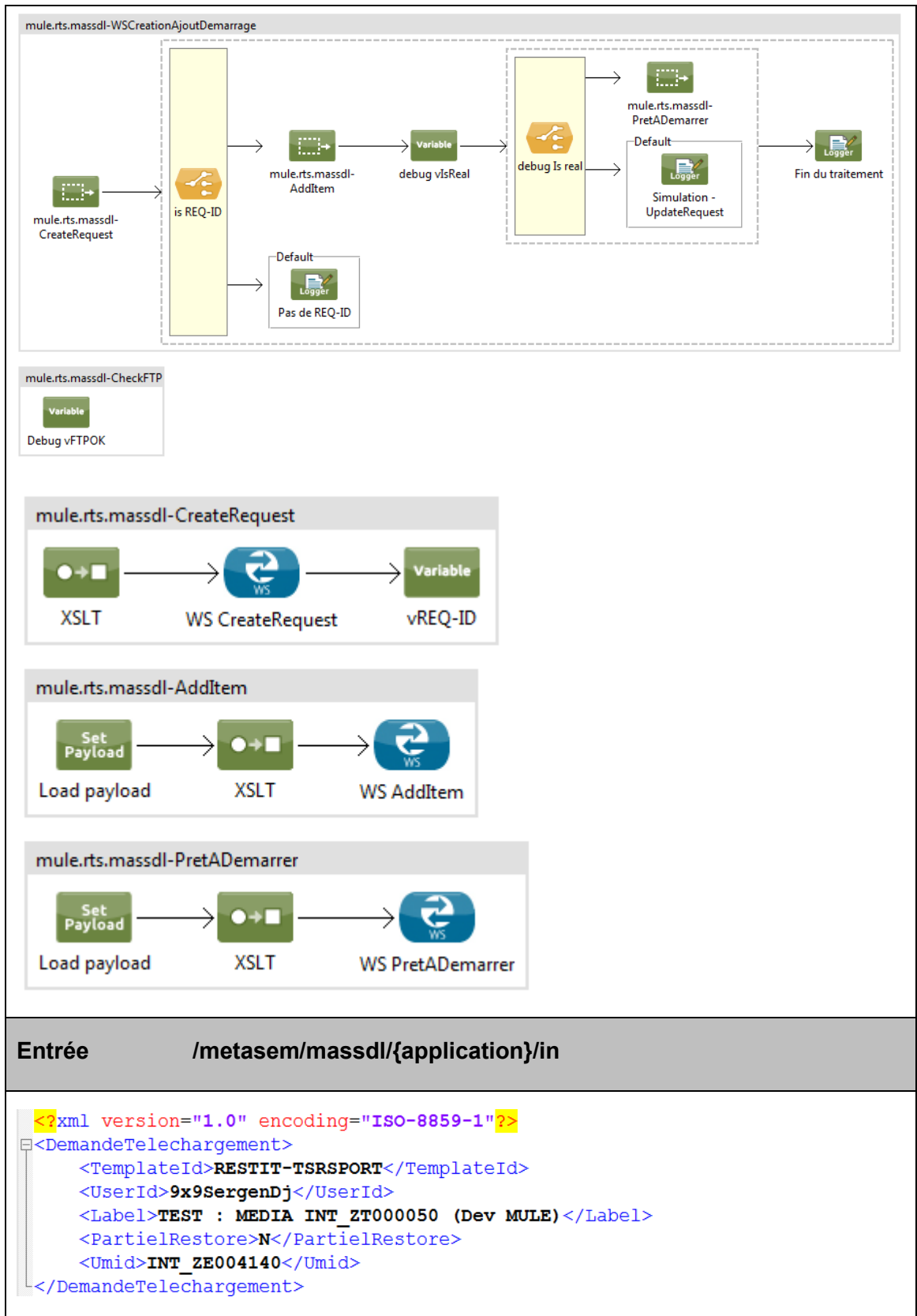


Figure 33 : Modèle Mule MassDL








```
<?xml version="1.0" encoding="ISO-8859-1"?>
<DemandeTelechargement>
  <TemplateId>RESTIT-TSRSPORT</TemplateId>
  <UserId>9x9SergentDj</UserId>
  <Label>TEST : MEDIA INT_ZT000050 - Partiel (Dev MULE)</Label>
  <PartielRestore>O</PartielRestore>
  <NomRestitution>I'M A TEST</NomRestitution>
  <Umid>INT_ZT000050</Umid>
  <TcIn>10:00:00:14</TcIn>
  <TcOut>10:05:00:00</TcOut>
</DemandeTelechargement>
```

Sortie Téléchargement GICO

N°	Type	Date création	Description	Avancement	Ech	Créateur	Application
REQ-202556	Télécharg. HIRES	05.05.2015 10:46	TEST : MEDIA INT_ZE004140 - Full (Dev MULE)	Non démarré		Sergent, Djavan (RTS)	massDL
REQ-202555	Télécharg. HIRES	05.05.2015 10:40	TEST : MEDIA INT_ZT000050 - Partiel (Dev MULE)	Non démarré		Sergent, Djavan (RTS)	massDL

Composant	Nom	Rôle
Connector Database	DB check charge totale	Vérifier la charge totale du serveur (archivages / téléchargements)
	DB check nb ingest	Vérifier le nombre d'archivages en cours
	DB check nb DL	Vérifier le nombre de téléchargements en cours.
Connector Web Service Consumer	WS CreateRequest	Créer la requête initiale
	WS AddItem	Ajouter les médias à la requête créée précédemment
	WS PretADemarrer	Démarrer le traitement de la requête
Filter Schema Validation	Validation partiel	Valide la restitution partielle avec xsd-partiel.xsd
	Validation full	Valide la restitution totale avec xsd-full.xsd
Flow Control Collection Splitter	Collection Splitter	Divise la structure en entrée pour boucler sur chaque élément

Analyse du prototype

Ce prototype, le plus complexe réalisé jusqu'ici, regroupe l'ensemble des notions apprises auparavant. On peut ainsi voir l'utilisation de composants tels que le connecteur *Database* ou le *Web Service Consumer* combiné au *Mule Requester*, au *Choice* aux *Variables* etc. De plus, les transformations XSLT permettent un traitement puissant de la fiche en entrée, malgré la grande difficulté d'apprentissage de ce langage

- + Utilisation de la quasi-totalité des composants vus jusque la
- + Utilisation de Web Service
- + Accès Base de données
- + Transformations XSLT
- + Filtres XSD

- Complexité de la représentation Mule
- Difficulté de maintenance
- Nombreuses variables
- Non terminé : FTP à gérer
- Gestion erreurs FTP/template à améliorer

6. Mise en production

Note : voir document *RTS_Exploitation-Mule_ESB.docx* pour le dossier d'exploitation

6.1 Fichiers de propriétés

Nous avons vu la méthode utilisée pour développer pour de multiples environnements aux propriétés distinctes avec les fichiers de propriétés. Lors du déploiement de l'application sur un serveur, si l'on veut que l'application se base sur les propriétés serveur, il faut modifier ce fichier. Premièrement, un seul fichier sera utilisé pour tous les environnements. Nous l'appellerons « *app.properties* ». Ce fichier contient toutes les informations et configurations qui ne dépendent pas de l'environnement. Dans un second temps, il faut créer le fichier « *mule-app-override.properties* » et l'ajouter au *Property Placeholder*. Ce fichier, situé hors de l'application sur le serveur, contiendra les informations relatives à l'environnement. Ainsi, une application déployée sur le serveur de test pointera toujours vers cet environnement sans risque d'incohérence.

Exemple :

```
1 baseDir=/data/${mule.env}/metasem
2 appDir=${baseDir}/ibetasuisse/lto_bab
```

Ces deux lignes proviennent du fichier *app.properties* de l'application Betasuisse. La variable *\${mule.env}* est, elle, définie dans le fichier *mule-app-override.properties*. En écrivant tous les fichiers d'applications de cette façon, toutes les applications déployées sur le serveur seront cohérentes vis-à-vis de l'environnement. Pour définir la variable d'environnement au lancement de l'application (si celle-ci n'est pas définie dans le fichier *mule-app-override.properties*) il faut exécuter ces commandes :

```
/.bin/mule -M-Dmule.env=production
$ mule -M-Dmule.env=production
```

6.2 Gestion d'erreurs

Dans les processus développés, il est extrêmement important que les fiches traitées ne disparaissent pas en cas d'erreur. Afin d'éviter ceci, un système de gestion d'erreurs a été mis en place. Celui-ci consiste à récupérer le payload d'origine (le fichier) dans une variable à laquelle on accèdera dès qu'une erreur est provoquée pour réécrire le fichier dans le dossier d'erreurs. En parallèle, un fichier de log est généré. Il contient l'erreur telle que Mule l'a décrite. Ces deux fichiers sont nommés selon la règle suivante :

- <nom_fichier>.extension → Fichier erreur
- <nom_fichier>.extension.log → Fichier de log

Afin d'alléger le modèle Mule, nous utiliserons des transformateurs globaux (voir ci-dessous) pour gérer la transition de payload.

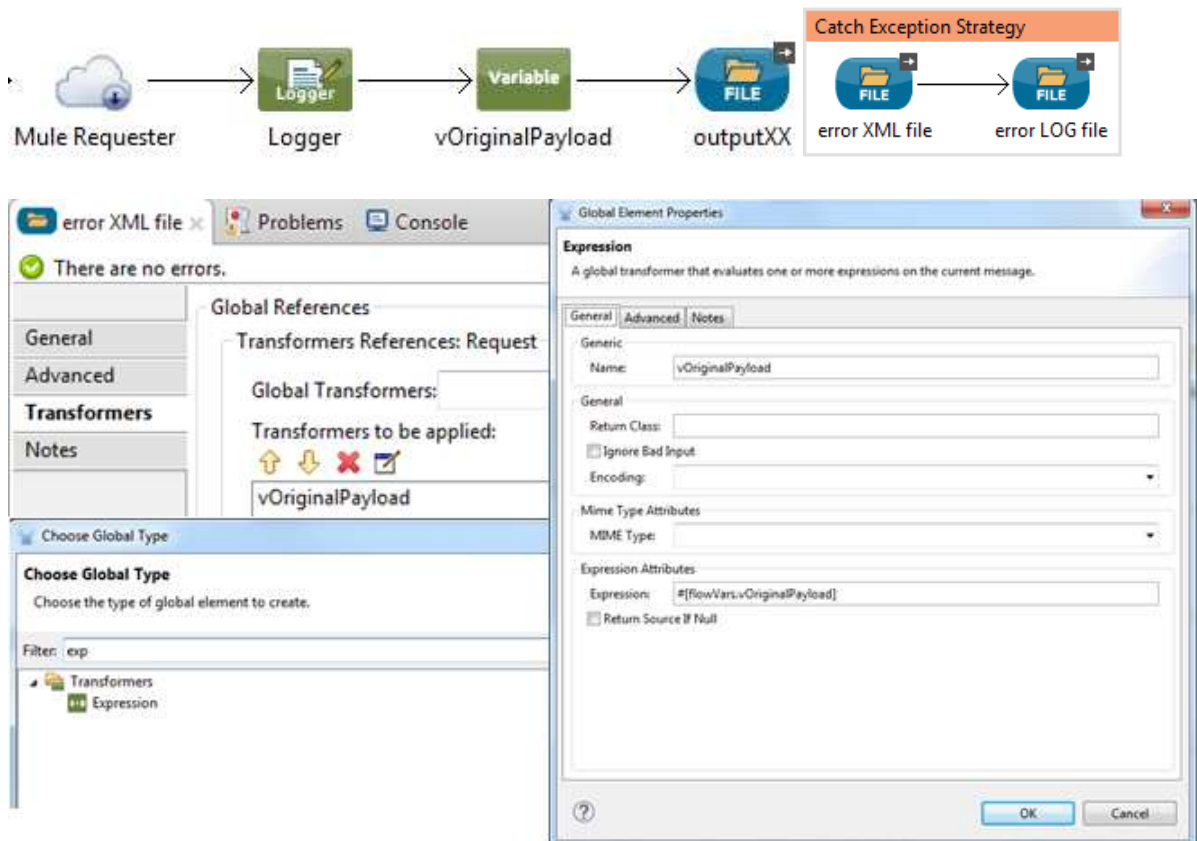
6.3 Transformateurs globaux

Mule offre la possibilité, pour certains de ses composants, d'appliquer un transformateur directement. Ce transformateur peut être de tous les types présents dans la palette. Nous utiliserons le type *Expression*, qui nous permet d'utiliser une expression pour définir la sortie. Un exemple d'utilisation :

- J'enregistre le fichier en entrée dans la variable *vOriginalPayload*
- Je définis un transformateur de type *Expression* dans le composant *File* situé dans le composant *Catch Exception Strategy*
- Je définis mon expression à « `#[flowVars.vOriginalPayload]` » afin d'accéder à la variable précédemment initialisée.

Ainsi, lors de chaque erreur, le composant *File* va écrire un fichier en sortie en lui appliquant le transformateur qui lui indique de charger le payload original. On pourrait, par exemple, ajouter une ligne au fichier pour indiquer ce qui a provoqué l'erreur ou toute autre information.

Figure 34 : Gestion d'erreurs et définition d'un transformateur global



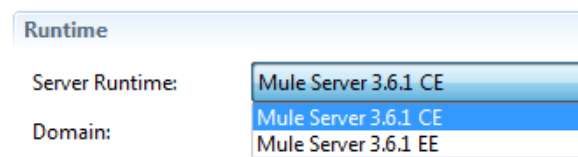
6.4 Runtime CE

Par défaut, le runtime¹² installé avec *Anypoint Studio* est la version entreprise. Afin d'être alerté des composants inutilisables en version *Community*, il faut installer le runtime correspondant. La procédure est décrite ici :

<http://www.mulesoft.org/documentation/display/current/Adding+Community+Runtime>

On peut ensuite changer de runtime à loisir. Il faut cependant rester attentif aux faits suivants : certains composants (par exemple FTP) n'ont pas tous le même type de configuration entre les différents runtime et provoquent des erreurs lors des changements. Par ailleurs, le mode Mule debugge qui permet de gérer son flux lors du débogage ainsi que de visualiser la valeur de chaque variable à n'importe quel moment du flux, n'est disponible que dans la version *Entreprise*. Il est donc intéressant de développer avec la version *Entreprise*, puis d'effectuer la transition vers la version *Community*. Pour changer, après installation, il suffit d'ouvrir le fichier *mule-project.xml* de l'application et de sélectionner le runtime désiré, CE ou EE.

Figure 35 : Changement de runtime



6.5 Monitoring

6.5.1 MMC

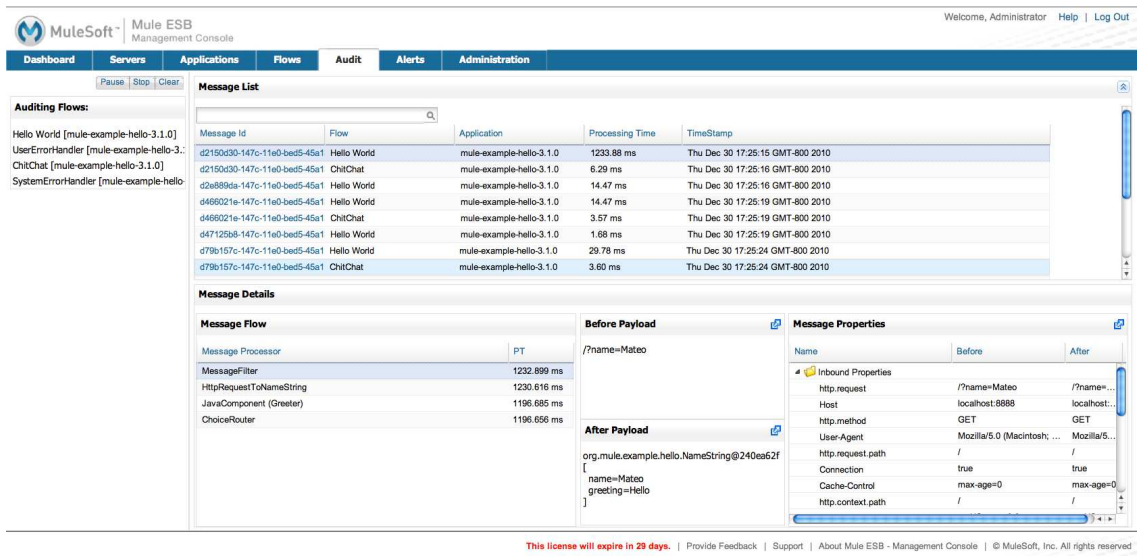
La console de Management Mule (Mule Management Console – MMC) permet, comme son nom l'indique, de gérer les flux. Cette console fonctionne pour tous les flux, quel que soit l'endroit où ils sont déployés. Il s'agit d'une interface web qui centralise la gestion des processus. Les fonctions clés de la MMC sont :

- Management centralisé
- Contrôle de flux plus fin
- Meilleure sécurité
- Diagnostique et statistiques
- Alertes intelligentes
- Gérance des clusters¹³
- Visualisation d'évènement

¹² Chargé de l'exécution de l'application, voir chapitre 10 - Glossaire

¹³ Grappe de serveurs, voir chapitre 10 - Glossaire

Figure 36 : Mule Management Console



Cette console se présente donc sous la forme d'une interface Web. Chaque flux peut être analysé, comparé, stoppé etc.

6.5.2 JMX

Java Management Extensions (JMX) est un ensemble de spécification définissant un agent¹⁴ Java. Ces spécifications concernent l'architecture, les services et l'API¹⁵. Cet agent permet d'effectuer des relevés de performances, de gérer et de surveiller une application Java lors de son exécution. Cela représente un standard pour le monitoring local ou distant et permet de notifier les dysfonctionnements et de modifier dynamiquement le comportement de l'application, par exemple en redémarrant un composant ou en modifiant les fichiers de configuration.

L'agent JMX est intégré à Mule et est très simple à mettre en place. Il suffit de créer un flux qui contient :

Entête :

```
[...]xmlns:management=http://www.mulesoft.org/schema/mule/management [...]
```

```
[...]http://www.mulesoft.org/schema/mule/management/3.2/mule-management.xsd[...]
```

Flux :

```
<management:jmx-server enableStatistics="true">
  <management:connector-server url="service:jmx:rmi:///jndi/rmi://localhost:9101/jmxrmi" />
</management:jmx-server>
<management:jmx-default-config port="9100" registerMx4jAdapter="true">
</management:jmx-default-config>
<management:jmx-log4j />
<management:jmx-mx4j-adaptor cacheXsl="false"
  jmxAdaptorUrl="http://${mule.applicationServerName}:9200">
</management:jmx-mx4j-adaptor>
```

¹⁴ Fournit un service, voir chapitre 10 - Glossaire

¹⁵ Application Programming Interface, voir chapitre 10 - Glossaire

Une fois cela effectué, on peut accéder aux statistiques et autres données récoltées via une interface web (<http://{host}:9200>) ou par une application dédiée, type *VisualVM*.

6.6 Le flux console

En créant un unique flux qui comporte le code susmentionné, il est possible de monitorer tous les flux exécutés sur le serveur.

De cette façon, on peut accéder aux statistiques via une connexion distante (RMI)¹⁶, une interface web ou les fichiers de logs.

Nom du flux	mule.rts.management-console
Contexte	Permet de disposer de statistiques pour tous les flux et de démontrer la possibilité de créer une interface d'administration personnalisée basée sur le service JMX.
Description	<p>Déclencheur : Connexion à http://{host}:9700</p> <p>Processus : Monitoring et gestion des flux.</p> <p>Résultat : Tableau HTML.</p> <p>Lors de la connexion au port http://{host}:9700, on récupère la liste des flux Mule en cours pour générer une liste. On appelle pour cela l'agent JMX sur le port 9200. Après quelques conversions, si le nom de flux commence par « Mule. » il est ajouté à la liste des flux à monitorer. On génère ensuite le début de la page HTML, auquel on ajoute le tableau des flux. Cette étape est réalisée en envoyant une requête particulière pour chaque flux en cours au port 9200 afin de récupérer les statistiques au format XML, qui sont ensuite converties en HTML via une transformation XSLT. Enfin, on finit la page HTML afin de la rendre valide et interprétable par le navigateur.</p>
Contraintes	-
Préconditions	L'agent JMX est activé et fonctionnel

¹⁶ Remote Method Invocation, voir chapitre 10 - Glossaire

Modèles

Figure 37 : Modèle UML Management-Console

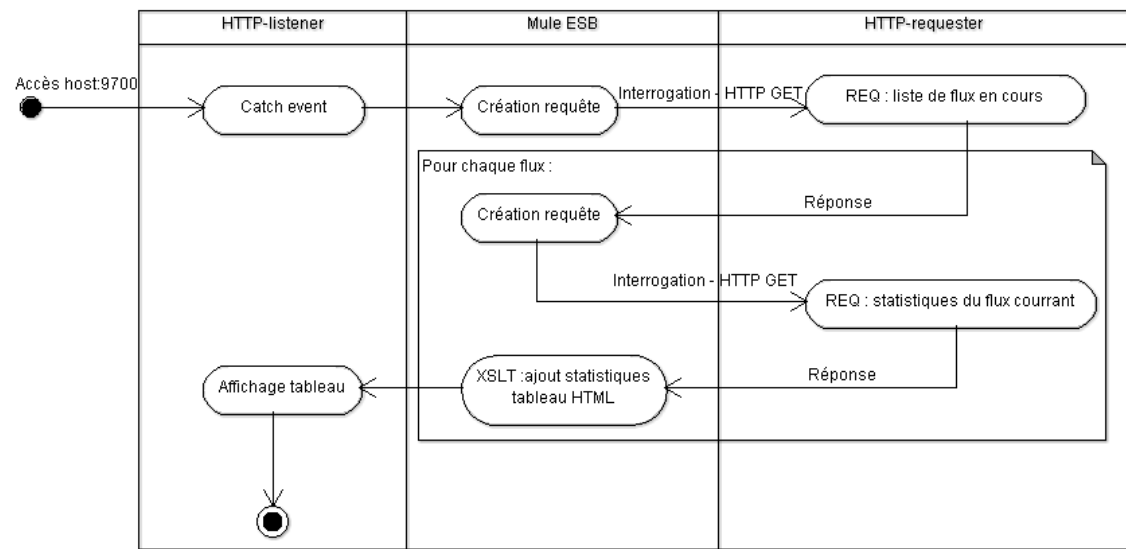
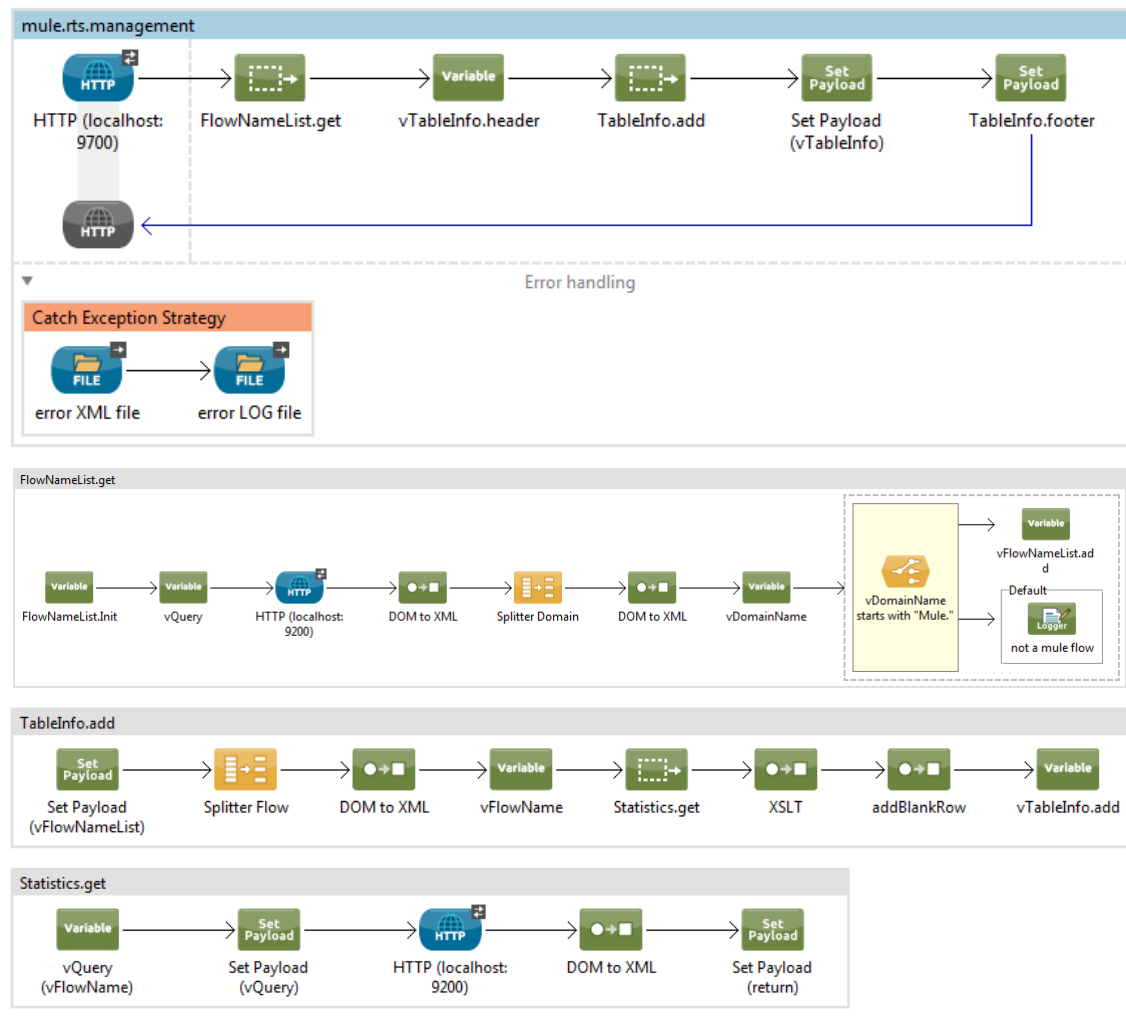
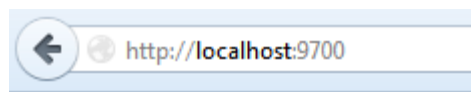


Figure 38 : Modèle Mule Management-Console



Entrée {host}:9700



Sortie {host}:9700

Mule RTS Management Console

Flux	Event async	Total event	Exception	Erreurs	Temps de traitement (avg)
mule.rts.xml-betabab	0	0	0	0	0
mule.rts.importrecursif-ExitPointOK	0	0	0	0	0
mule.rts.importrecursif-ExitPointKO	0	0	0	0	0
mule.rts.importrecursif-EntryAllApp	0	0	0	0	0
mule.rts.management	0	4	0	0	621
mule.rts.massdl	14	14	0	0	344
mule.rts.moveandrename-EntryExemple2	0	0	0	0	0
mule.rts.moveandrename-ExitPointOK	0	0	0	0	0
mule.rts.moveandrename-EntryExemple3	0	0	0	0	0
mule.rts.moveandrename-ExitPointKO	0	0	0	0	0
mule.rts.moveandrename-EntrySupportStatus	0	0	0	0	0
mule.rts.rtsinfochip	0	0	0	0	0
mule.rts.sparkex-AcquittementToDam	0	0	0	0	0
mule.rts.sparkex-ErrorToDam	0	0	0	0	0
mule.rts.sparkex-DemandeToSparkex	0	0	0	0	0
mule.rts.sparkex-ArchivesToDam	0	0	0	0	0
mule.rts.thumbfile	0	0	0	0	0
mule.rts.thumbfile-requester	13	13	0	0	1

Composant	Nom	Rôle
Splitter	Splitter Domain	Expression XPATH qui génère une ligne par domaine (liste des flux).
	Splitter Flow	Expression XPATH qui génère une ligne par flux (statistiques de chaque flux).
Transformateur	DOM To XML	Obligatoire pour utiliser le résultat du Splitter et du composant HTTP.

	XSLT	Génère du code HTML qui est renvoyé au composant HTTP pour être affiché
Connecteur HTTP	HTTP (localhost:9700)	Port d'écoute pour générer la console.
Analyse du prototype		
<p>Ce prototype permet d'obtenir des informations pour chaque flux et a permis de mettre en place une interface de monitoring très basique. De plus, nous avons pu voir l'utilisation du composant Splitter qui permet de boucler sur les balises d'un fichier XML. Il faut cependant faire bien attention, car le splitter nécessite d'être placé au bon endroit pour fonctionner correctement, la plupart du temps dans un sous flux de la taille de la boucle à effectuer.</p>		
<ul style="list-style-type: none"> + Génération de code HTML + Monitoring personnalisé + Utilisation agent JMX gratuit + Splitter pour boucler sur un fichier XML + Tableau dynamique des flux en cours 		<ul style="list-style-type: none"> - Aucune méthode (start, stop etc.) - Interface basique

7. Evaluation des prototypes

Compilation : Tous les prototypes sont compilables.

7.1 Tests

Tableau 3 : Tableaux récapitulatifs des tests

RTSINFOCHIP			
Entrée	Résultat obtenu	Statut	Commentaire
Fichier TXT	Fichier dans dossier KO	OK	Seuls les fichiers XML doivent être pris en compte.
Fichier XML	Fichier traité selon balise status	OK	Voir ci-dessous pour le routage
Balise status = running	Fichier déplacé dans OK	OK	Historique
Balise status = success	Fichier déplacé dans OK	OK	Traitement standard si toutes les conditions sont remplies.
	Transformation XSLT	OK	
	Dépôt sur FTP	OK	
Balise status = error	Fichier déplacé dans KO + log	OK	Historique + log
Fichier dont la balise status n'existe pas	Catch Exception Strategy	OK	Historique + log
Transformation XSLT	Fichier transformé	OK	xsl-pattern.xsl
Erreur	Catch Exception Strategy		Historique + log

THUMBFIL			
Entrée	Résultat obtenu	Statut	Commentaire
Fichier TXT	Fichier dans dossier KO	OK	Seuls les fichiers XML doivent être pris en compte
Fichier XML	Fichier traité	OK	
1 fichier XML Places restantes : 0	Processus boucle, essai / 5s	OK	Une fois le fichier récupéré, le processus tourne tant qu'aucune cible n'est définie.
1 fichier XML Places restantes : 1	Fichier déplacé	OK	Dossier le moins remplis
2 fichiers XML Places restantes : 1	1 fichier déplacé, 1 fichier boucle	OK	
5 fichiers XML Places restantes : 15	5 fichiers déplacés	OK	
Erreur	Catch Exception Strategy	OK	Historique + fichier de log

BETASUISSE JSON			
Entrée	Résultat obtenu	Statut	Commentaire
Fichier TXT	Fichier non pris en compte	OK	Seuls les fichiers XML doivent être pris en compte.
Fichier XML	Fichier traité	OK	
Balise sMediaID	Balise récupérée,	OK	si absente fichier KO

Structure JSON du composant HTTP	Structure LinkedHashMap, pas JSON	KO	Problème de communication entre composants. JSOToXML ne fonctionne pas.
Génération de fichier	Fichier généré	OK	Via classe Java -> solution non définitive
Class FileGenerator	Classe OK, fichier généré	OK	Tous ces traitements doivent être fais dans Mule, pas via les classes Java. Voir Betasuisse XML pour la seconde version.
Class BabAnalyzer	Classe OK, gère le processus	OK	
Class PropertiesFileReader	Classe OK, lis les propriétés du fichier	OK	
Erreur	Catch Exception Strategy	OK	Historique + fichier log

BETASUISSE XML			
Entrée	Résultat obtenu	Statut	Commentaire
Fichier TXT		OK	Seuls les fichiers XML doivent être pris en compte
Fichier XML		OK	
Balise sMediaID		OK	Si absente fichier KO
Structure XML du composant HTTP	Structure OK	OK	Composant DOMToXML fonctionne
Transformation	Fichier	OK	solrResponse_add_str_index.xsl

XSLT		transformé		solrResponse_indexed_to_gicoStock.xsl
Génération de fichier		Fichier déposé	OK	Via composants Mule -> OK
Choice nombre résultats	sur de	Si > 1 KO Si = 1 OK Si < 1 = KO	OK	Routage pour gestion d'erreurs
Erreur		Catch Exception Strategy	OK	Historique + fichier de log

MOVE AND RENAME			
Entrée	Résultat obtenu	Statut	Commentaire
/app/type/in/fichier.xml	/sonaps/import_news/in #app#type#fichier.xml	OK	-
/sonaps/import_news/ok/ #app#type#fichier.xml	/app/type/ok/fichier.xml	OK	-
/sonaps/import_news/ko/ #app#type#fichier.xml	/app/type/ko/fichier.xml	OK	-
Erreur	Catch Exception Strategy	OK	Historique + log

IMPORT RECURSIF			
Entrée	Résultat obtenu	Statut	Commentaire
/import-in/app/type/in/ fichier.xml	/import-result/[ok ko]/ #app#type#[ok ko]#fichier.xml	OK	-

/sonaps/import_result/ok/ #app#type#fichier.xml	/app/type/ok/fichier.xml	OK	-
/sonaps/import_result/ko/ #app#type#fichier.xml	/app/type/ko/fichier.xml	OK	-
Erreur	Catch Exception Strategy		Historique + log

THUMBFILEREQUESTER			
Entrée	Résultat obtenu	Statut	Commentaire
Fichier TXT	Fichier dans dossier KO	OK	Seuls les fichiers XML doivent être pris en compte
Fichier XML	Fichier traité	OK	-
1 fichier XML Places restantes : 0	Log pas de place disponible, fin du processus	OK	Processus libéré si pas de place disponible
1 fichier XML Places restantes : 1	Fichier déplacé	OK	Dossier le moins remplis
2 fichiers XML Places restantes : 1	1 fichier pris en charge et déplacé, 1 fichier en attente du prochain tour de processus	OK	Processus 1 fiche 1 place suivi du processus 1 fiche 0 places
5 fichiers XML Places restantes : 15	5 fichiers déplacés	OK	1 fiche déplacée toutes les 20s (poll)

0 fichiers XML	Log aucun fichier à traiter	OK	-
Erreur	Catch Exception Strategy	OK	Historique + log

SPARKEK			
Entrée	Résultat obtenu	Statut	Commentaire
Fichier entrant sur DAM	Fichier déplacé sur SPARK	OK	-
Fichier entrant sur SPARK	Fichier déplacé sur DAM	OK	-
Erreur	Catch Exception Strategy	OK	Historique + log

MASSDL			
Entrée	Résultat obtenu	Statut	Commentaire
Fiche XML Nb DL & ingest < limite	Vérification charge totale OK, vérification nombre de fichiers en attente	OK	-
Fiche XML Nb DL & ingest > limite	Vérification charge totale KO, vérification nombre de fichier en attente	OK	-
0 fichier XML Nb DL & ingest < limite	Log aucun fichier à traiter	OK	-
1 fichier XML Nb DL & ingest < limite	Récupération du fichier	OK	-
1 fichier XML	Vérification de la	OK	-

Nb Template < limite	cohérence du fichier		
Vérification cohérence du fichier	Filtre XSD	OK	Vérifie l'intégrité et la complétude du fichier.
Fichier vérifié OK	Création requête WS	OK	-
Fichier vérifié KO	Log fichier invalide	OK	-
REQ-ID créé (WS CreateRequest)	WS AddItem	OK	Résultat du WS
REQ-ID non créé	Erreur	OK	Catch Exception Strategy
Item ajoutés (WS AddItem)	WS PretADémarrer	OK	-
Lancement du téléchargement (WS PretADémarrer)	Téléchargement lancé	OK	-
Erreur	Catch Exception Strategy	OK	Historique + log

Note : Les deux liens ci-dessous permettent de valider et tester le code XSLT

http://www.utilities-online.info/xslttransformation/#.VQ_jmi6X-pA

<http://xslttest.appspot.com/>

7.2 Récapitulatif des composants utilisés

Le tableau suivant récapitule pour la plupart des composants rencontrés leurs avantages et inconvénients. Une case vide ne veut pas dire que le composant est exempt d'avantages/inconvénient mais simplement que ceux-ci n'ont pas été mis en avant lors de la réalisation des prototypes. Il s'agit là d'un avis subjectif basé sur la conception et la réalisation du projet dans son ensemble.

Tableau 4 : Récapitulatif des composants utilisés

Composant	Avantage	Inconvénient
File	Bonnes fonctions Facile d'utilisation Nombreux paramètres	Doit être le premier élément du flux pour se comporter en source Prend tous les fichiers présents dans le dossier (sauf filtre filename)
FTP	Facile d'utilisation	Version CE différente de la version EE
HTTP	Facile d'utilisation Bonnes fonctions Mis à jour récemment	Mauvaise maîtrise du format de retour.
Database	Bonnes fonctions Nombreux paramètres Nombreux SGBD supportés	Mauvaise maîtrise du format de retour.
Web Service Consumer	Facile d'utilisation Compatible DataSense ¹⁷	Mauvaise gestion des erreurs, debug difficile.
Poll	Facile d'utilisation	1 seul composant à l'intérieur, restreint les traitements
Flow Reference	Très utile pour clarifier un flux	-
Java	Bonnes fonctions Permet d'étendre les fonctions de Mule.	Apprentissage difficile au début pour comprendre comment manipuler les données de Mule.

¹⁷ Permet l'analyse des métadonnées, voir chapitre 10 - Glossaire

	Intégrable aux variables	
Logger	Facile d'utilisation Très utile au debugge	-
Mule Requester	Facile d'utilisation Performant Permet la lecture d'un fichier en cours de flux	Priorité fichiers non gérée AutoDelete du fichier d'entrée automatique Peu de paramètres
Expression	Large éventail de possibilité avec MEL	-
Variable	Beaucoup de possibilités d'initialisation Facilité d'accès aux données Sauvegarde du payload	1 boîte variable par variable dans l'application, surcharge visuelle du flux
XSLT	Facile d'utilisation Performant	
DOMToXML	Performant	
FileToString	Performant	
JSONToXML	-	Inutilisable car ne prend qu'une partie du JSON reçu. Non performant
DataMapper	Convertit n'importe quelle source de données en une autre Facile d'utilisation	Payant Boite obscure dont on ne maîtrise aucun aspect.

	<p>Performant</p> <p>Simplifie la mise en place d'un flux</p>	
Set Payload	Performant	Faible paramétrage
Filter	Large palette de filtres	-
Choice	Facile d'utilisation	Mauvaise gestion des erreurs, messages peu compréhensible.
Collection Splitter	<p>Facile d'utilisation.</p> <p>Particulièrement adapté aux connecteurs Database</p>	-
Scatter-Gather	<p>Duplique un message et permet des traitements parallèles.</p> <p>Facile d'utilisation</p>	<p>Ordre des opérations non optimal</p> <p>Accès aux variables non optimal</p>
Round Robin	Facile d'utilisation	Faible paramétrage
Catch Exception Strategy	Permet une gestion des erreurs basique mais simple à mettre en place.	Peu d'informations sur la mise en place de la gestion d'erreur.

7.3 Problèmes rencontrés

Lors de la réalisation de ces prototypes, plusieurs problèmes ont été relevés.

La plupart du temps, il s'agit de problèmes liés au paramétrage des composants (Database en Dynamique plutôt que Parametrized), des fonctions manquantes (Order processing pour le Mule Requester) ou aux formats de données (Betasuisse JSON). Les prototypes sont orientés pour faire un maximum du travail avec les composants présents dans la version Community de Mule, et ajouter la logique dans les variables afin d'avoir un flux lisible et maintenable.

D'autres fois, il s'agissait de manque de lisibilité des messages d'erreurs et de la difficulté de les interpréter y compris en mode debugge. Autre facteur, l'instabilité de Mule suite à sa propre mise à jour et de celle de son Workspace.

Les points suivants ont été mis en avant par les prototypes :

- Mule manipule très bien les XML mais ne gère pas le format JSON correctement.
- Afin de récupérer un fichier en cours de flux (après vérifications de l'état du système par exemple) il faut étendre Mule et installer le composant *Mule Requester*, car Mule ne prend pas cette fonctionnalité en charge nativement.
- Si l'on veut utiliser le composant Java pour faire un traitement automatique dans un flux, il faut placer la méthode appelée dans la méthode `onCall()` de la classe appelée par le composant Java. C'est cette méthode qui est appelée par le flux au moment du passage dans le composant. Les méthodes qui retournent un résultat (String, Integer ...) peuvent être utilisées pour initialiser une variable ou pour router un message.
- Pour qu'un flux n'exécute pas plusieurs traitements parallèles, il faut définir la stratégie à synchrone (`synchronous`) dans le flux principal de l'application concernée. Ainsi, le flux ne commence pas une nouvelle tâche avant d'avoir fini l'action en cours.
- Pour définir un environnement, il faut initialiser une variable d'environnement et configurer un fichier de propriétés (`.properties`) contenant les paramètres.
- Pour déployer une application, il faut la déposer sous un format d'archive déployable dans un dossier spécifique de Mule afin que le logiciel la prenne en charge. Ne pas oublier de spécifier les variables d'environnement non définies dans les fichiers `app.properties` et `mule-app-override.properties`

8. Conclusion

Ce projet nous aura permis d'analyser *Mule ESB 3.6.1* et on peut en déduire que :

Fonctions et composants

Mule ESB présente de bonnes fonctions mais est relativement limité dans sa version *Community*. De plus, certains composants ne se comportent pas de la même façon entre les deux versions (EE ; CE). Enfin, les composants ne sont pas clairement indiqués comme faisant partie de l'une ou l'autre version.

Stabilité du logiciel

Mule ESB présente une bonne stabilité dans sa version *Standalone*, mais a du être réinstallé plusieurs fois au cours de ce projet. En effet, les mises à jour du logiciel provoquent parfois une incompatibilité avec le Workspace courant, auquel cas il faut en re-générer un et importer chaque projet à l'aide d'archives déployables. Parfois, c'est le debugger qui nécessite un redémarrage de Mule (sans réinstallation). A noter que Mule ne s'installe pas à proprement parler, il suffit donc d'avoir une copie saine de backup pour retrouver un système optimal en la copiant.

Documentation

Sur le site officiel, la documentation est relativement complète et illustrée. Cependant, la plupart des exemples ne vont pas assez loin et il est souvent nécessaire de se tourner vers d'autres sites d'aide (comme *stackoverflow.com* par exemple) pour trouver un exemple de réalisation complet.

Mule propose une fonction de génération automatique de documentation pour les flux. Cette documentation au format HTML résume chaque flux et sous flux avec son code spécifique, son modèle Mule ainsi que la description détaillée de chaque composant que l'on aura commenté.

Interface

L'interface de développement de Mule, *Anypoint Studio*, présente de nombreux avantages comme la conception rapide, le drag-and-drop de composants, leur paramétrage, les fonctions de débogage des applications ainsi que les différentes vues (Flow et XML). On peut également la personnaliser à loisir. Cette interface, intégrée à Eclipse, est intuitive et facile à utiliser.

Marché

Le marché des ESB dispose d'offres complètes autant Open source que propriétaire. Bien que *Mule ESB* soit un acteur reconnu et disposant d'une grande notoriété, d'autres tels que *WSO2* ou *Talend* sont également très souvent cités par les utilisateurs comme étant d'excellentes alternatives (non testées durant ce projet).

Modélisation

Le logiciel de modélisation *ArgoUML* dispose de bonnes fonctionnalités mais est peu tolérant aux mises-à-jour de diagrammes (mise en page réinitialisée presque systématiquement en cas de suppression d'un élément). Cependant, les diagrammes réalisés ici sont de haut niveau et ne comportent pas beaucoup d'étapes, ce qui simplifie la modélisation des flux.

Tests

Chaque prototype a été testé afin de disposer d'une base pour les tests de non-régression. De plus, ils disposent tous d'un système de gestion d'erreurs qui prévoit la sauvegarde de la fiche traitée et la génération d'un fichier de log en cas d'imprévu. Enfin, tous les prototypes sont compilables et utilisables.

Déploiement

Il est facile de déployer une application dans Mule en prenant en considération son environnement. Il suffit pour cela de déposer une archive de l'application dans un dossier spécifique pour que Mule la prenne en charge.

Il est également possible de démarrer, stopper et mettre à jour une application sans provoquer de conflits avec les autres applications présentes sur le serveur.

Management

Les flux peuvent être administrés depuis l'interface web proposée par JMX (également via un logiciel comme *VisualVM*). Cette interface permet d'accéder à un ensemble de statistiques (nombre d'évènements traités, temps moyen, nombre d'erreurs, point d'entrée etc.) et de générer un fichier CSV, HTML ou XML qui résume ces propriétés. Pour disposer de cette console d'administration, il suffit de lancer le projet *mule.rts.management-console* qui met à disposition les statistiques de tous les flux actifs et de se rendre à l'adresse <http://{host}:9700> afin d'accéder à la console personnalisée ou à l'adresse <http://{host}:9200> pour la console par défaut de Mule-JMX.

9. Evaluation personnelle

Ce projet m'aura apporté beaucoup de connaissances dans un domaine qui n'a pas été étudié à la HEG, les bus de services d'entreprise. C'est en effet un système très intéressant qui permet à des systèmes hétérogènes de communiquer, et cela n'avait pas été vu durant ma formation.

J'ai, par exemple, eu l'occasion d'effectuer de nombreuses transformations XSLT afin d'obtenir un fichier XML différent ou de générer un fichier HTML, ce qui a permis la création d'une interface d'administration (basique). J'ai également eu la possibilité, au travers des nombreuses problématiques posées par les prototypes, de renforcer mes connaissances en programmation Java et XML. De même pour mes compétences techniques en organisation, analyse, conception, réalisation et modélisation de prototypes mais aussi sur le déploiement et le monitoring de ceux-ci.

J'ai également appris à connaître l'infrastructure RTS, les collaborateurs et le parc applicatif. J'ai eu l'occasion de voir comment, de la cassette physique contenant l'archive jusqu'à la diffusion, le média était stocké, transporté et mis à disposition. Ce fut également le cas pour certains outils tels que Gico, Solr ou encore Sonaps.

Ce travail m'aura également permis de mener à bien un projet du début à la fin en respectant un cahier des charges, un planning, des priorités et d'autres contraintes liées aux prototypes tout en étant proactif dans le développement et en ayant une approche orientée solutions.

D'un point de vue humain, ce stage m'aura aussi permis de faire la connaissance de l'équipe broadcast et de faire la très intéressante visite de la régie RTS durant la diffusion du télé-journal.

10. Glossaire

Agent Java	Propose des services et une API
AMQP	Advanced Message Queuing Protocol, protocole standardisé pour les messages intergiciels.
Anypoint Studio	Environnement de développement de Mule
API.....	Permet à un logiciel de demander des services à un autre via des classes, des méthodes et des fonctions standardisées.
Chaîne IP.....	Diffuse les médias sur le site web de la RTS
Cluster	Grappe de serveurs (ferme de calcul, minimum 2 serveurs) partageant la plupart du temps une ou plusieurs baies de disques.
DataSense.....	Fonctionnalité de traitement des métadonnées qui permet de connaître le type et la structure des données attendues par les composants.
Diagramme d'activité.....	Modèle UML utilisé pour décrire un workflow, le déclenchement d'un processus en fonction de l'état du système, la gestion de threads parallèles etc. Correspond la plupart du temps à l'algorithme du processus.
Eclipse	Environnement de développement libre, base d'Anypoint Studio
ESB	Architecture logicielle permettant aux applications hétérogènes de communiquer
MEL	Mule Expression Language, langage d'évaluation d'expressions de Mule
Payload	Objet contenant les données manipulées par un flux dans un bus d'entreprise.
RMI.....	Remote Method Invocation, API Java permettant l'exécution de commandes distantes.
Runtime	Environnement d'exécution chargé d'assurer les fonctions d'exécution d'application (gestion des entrées/sorties, erreurs, accès services OS etc.)
XPATH	Langage de positionnement dans un code XML
XSLT.....	Langage de transformation sur les fichiers XML, dans des fichiers à l'extension « .xsl »

Bibliographie

Documentation de base (dernière consultation le 16.04.15) :

<http://www.mulesoft.org/documentation/display/current/Home>

Tutoriel de prise en main (dernière consultation le 03.03.15) :

<http://www.mulesoft.org/documentation/display/current/First+30+Minutes+with+Mule>

<http://www.mulesoft.org/documentation/display/current/First+Day+with+Mule>

<http://www.mulesoft.org/documentation/display/current/First+Week+with+Mule>

Notes de versions (dernière consultation le 03.03.15) :

<http://www.mulesoft.org/documentation/display/current/Mule+3.5.0+Andes+Release+Notes>

<http://www.mulesoft.org/documentation/display/current/Mule+ESB+3.6.0+Release+Notes>

Concepts basiques de Mule (dernière consultation le 03.03.15) :

<http://www.mulesoft.org/documentation/display/current/Mule+Concepts>

<http://www.mulesoft.org/documentation/display/current/Mule+Fundamentals>

<http://www.mulesoft.org/documentation/display/current/Mule+Message+Structure>

<http://www.mulesoft.org/documentation/display/current/Mule+Expression+Language+Basic+Syntax>

<http://www.mulesoft.org/documentation/display/current/Anypoint+Studio+Essentials>

Références des composants de transport (dernière consultation le 03.04.15) :

<http://www.mulesoft.org/documentation/display/current/File+Transport+Reference>

<http://www.mulesoft.org/documentation/display/current/HTTP+Transport+Reference>

<http://www.mulesoft.org/documentation/display/current/Database+Connector>

Glossaire (dernière consultation le 03.04.15) :

<http://www.mulesoft.org/documentation/display/current/Glossary>

DataSense (dernière consultation le 03.03.15) :

<http://www.mulesoft.org/documentation/display/current/DataSense>

<http://www.mulesoft.org/documentation/display/current/DataSense+Query+Editor>

Déploiement environnements multiples (dernière consultation le 03.04.15) :

<http://www.mulesoft.org/documentation/display/current/Deploying+to+Multiple+Environments>

Management (dernière consultation le 16.04.15)

<http://fr.wikipedia.org/wiki/JMX>

<http://www.mulesoft.org/documentation/display/current/Mule+Management+Console>

<http://www.oracle.com/technetwork/java/javase/tech/javamanagement-140525.html>

<http://www.mulesoft.org/documentation/display/current/The+Mule+Agent>

<http://www.mulesoft.org/documentation/display/current/JMX+Service>

<http://www.mulesoft.org/documentation/display/current/JMX+Management>

<http://stackoverflow.com/questions/856881/how-to-activate-jmx-on-my-jvm-for-access-with-jconsole>

<http://www.jmdoudoux.fr/java/dej/chap-jmx.htm>

<http://skebir.developpez.com/tutoriels/java/jmx/>

<http://www.oracle.com/technetwork/java/javase/tech/best-practices-jsp-136021.html>

<http://blog.xebia.fr/2008/05/02/java-agent-instrumentez-vos-classes/>

http://mulesoft.github.io/mule-agent/#_how_to_extend_mule_agent

http://mulesoft.github.io/mule-agent/#_adding_new_jmx_publisher

Définition d'un ESB (dernière consultation le 03.03.15) :

http://fr.wikipedia.org/wiki/Enterprise_service_bus

http://en.wikipedia.org/wiki/Enterprise_service_bus

Modélisation (dernière consultation le 03.03.15) :

http://fr.wikipedia.org/wiki/UML_%28informatique%29

<http://fr.wikipedia.org/wiki/ArgoUML>

Programmation (dernière consultation le 06.03.15) :

<http://stackoverflow.com/>

http://fr.wikipedia.org/wiki/Extensible_Stylesheet_Language_Transformations

<http://aruntechtalk.blogspot.ch/2013/06/start-stop-mule-3-flow-as-needed.html>

<http://xslttest.appspot.com/>

Etude de marché (dernière consultation le 05.03.15) :

<http://www.openlogic.com/events/on-demand-webinars/comparison-open-source-software-esb>

http://en.wikipedia.org/wiki/Mule_%28software%29

<http://esbperformance.org/display/comparison/ESB+Performance>

<http://www.infoq.com/articles/ESB-Integration>

<http://www.open-source-guide.com/Solutions/Developpement-et-couches-intermediaires/Esb-eai>

<http://blogs.mulesoft.org/to-esb-or-not-to-esb/>

<http://erik.doernenburg.com/2009/07/making-esb-pain-visible/>

<http://www.zdnet.com/article/dont-use-an-esb-unless-you-absolutely-positively-need-one-mule-cto-warns/>

Smile Open Source Solution. *ESB - Les meilleures solutions open source*. Septembre 2014. Disponible ici : <http://www.developpez.com/actu/75999/Enterprise-Service-Bus-ESB-les-meilleures-solutions-open-source-telechargez-gratuitement-le-nouveau-livre-blanc-de-Smile/>

Annexe 1 : Tableau comparatif des ESB

GENERAL	ESB	OPEN SOURCE				PROPRIETAIRE	
		MULE ESB	TALEND	FUSE	WSO2	BizTalk	WebSphere
	Langage	JAVA	JAVA	JAVA	JAVA	Visual Studio / .NET	JAVA
Axes d'analyse	Modèle économique	Libre : ESB de base Payant : licence commerciale, formations, expertise, support, abonnement services (cloud)	Libre : Tous les produits de base Payant : licence commerciale, formations, expertise, support	Libre : Tous les produits Payant : Formation, expertise, support	Libre : Tous les produits Payant : Formation, expertise, support, certification	Payant	Payant
	Spécificité du produit	Anypoint Platform	Talend Unified Platform	Intégration à l'écosystème Jboss, FUSE SW	Middleware orienté SOA	ok	ok
	Outil de développement	Anypoint Studio (Eclipse)	Talend Open Studio (Eclipse) Accès facile à la doc depuis le dev	Jboss Developer Studio (Eclipse)	WSO2 Developer Studio, ne permet pas les tests	ok	ok
	Web Service et connecteurs	Apache CXF, collaboration avec SAP, Cisco et Salesforce	Apache CXF, collaboration avec SAP, Salesforce, Solr	Apache CXF	Peu de connecteurs, bonne prise en charge de web service	ok	ok
	Médiation et routage	Large prise en charge, MEL	Large prise en charge, extensible, Apache Camel	Large prise en charge, Apache Camel	Large prise en charge, Synapse	ok	ok
	Sécurité	Large prise en charge	Large prise en charge	Propose JAAS	Large prise en charge	ok	ok
	Monitoring	Libre : JMX Payant : MMC	Libre : JMX Payant : TAC, Analyse de log	Libre : Hawtio (JMX)	Libre : JMX, SNMP	ok	ok
	Clustering	2-8 nœuds	Apache Karaf	Apache Karaf	-	ok	ok
	Performance	-	-	-	-	ok	ok
	Cloud	Offre payante, principal axe de développement par Mulesoft	En développement	Version Alpha	En développement	ok	ok
	Documentation	Excellente qualité	Qualité moyenne	Qualité moyenne	Bonne qualité	ok	ok
	Communauté	Très active	-	Très active	-	ok	ok
Prise en main	Assez facile (tutoriels et autre)	Difficile, large écosystème	Difficile, large écosystème	Difficile, large écosystème	ok	ok	
Gestion	BAM (Monitoring)	Payant (MMC)	Payant (TAC, analyse de log)	Libre (Hawtio)	Libre	Inclus	Inclus
	BRE (Rules engine)	Module complémentaire (Drools)	Module complémentaire (Drools)	Inclus (Drools)	Module complémentaire (Drools)	Inclus	Inclus
	BPM (Management)	Module complémentaire (JBPM)	Module complémentaire (Bonita Solution)	Inclus (JBPM)	WSO2 Business Process Server	Inclus	Inclus
FORCE	Documentation Léger et robuste Modulaire / Extensibilité Grande communauté active Large prise en charge des formats Facilité d'installation Facilité de prise en main Flux déjà en production	Écosystème Talend Bonnes fonctionnalités d'entreprise Module BPM complet Outils de développement Modulaire / Extensibilité	Écosystème JBOSS Bonnes fonctionnalités d'entreprise Version drools et jbpn supportée > autres esb Bonne communauté	Bonnes fonctionnalités d'entreprise Bonne documentation Écosystème WSO2	Support Haut taux de fonctionnalités entreprise Gestion de la montée en charge	Support Haut taux de fonctionnalités d'entreprise Gestion de la montée en charge	
FAIBLESSE	Composants entreprise payants Pricing compliqué	ETL orienté ESB Poids global Difficulté de mise en place Exigences matérielles Pricing compliqué	Difficile à prendre en main Qualité de la documentation	Pas de tests immédiats Difficile à prendre en main	Prix Mise en place difficile Courbe d'apprentissage Extensibilité / Maintenance Avantages sur l'Open source ? Et Open source payant ?	Prix (>200'000.-) Mise en place difficile Courbe d'apprentissage Extensibilité / Maintenance	

Annexe 2 : Console Management JMX

Domain: Mule.mule.rts.betasuisse-deploy			
Mule.mule.rts.betasuisse-deploy:name=Configuration	org.mule.module.management.mbean.MuleConfigurationService	Information on the management interface of the MBean	Unregister
Mule.mule.rts.betasuisse-deploy:name=MuleContext	org.mule.module.management.mbean.MuleService	Information on the management interface of the MBean	Unregister
Mule.mule.rts.betasuisse-deploy:name=Mx4jHttpAdapter	mx4j.tools.adaptor.http.HttpAdaptor	Information on the management interface of the MBean	Unregister
Mule.mule.rts.betasuisse-deploy:type=Application,name="application_totals"	org.mule.module.management.mbean.ApplicationService	Information on the management interface of the MBean	Unregister
Mule.mule.rts.betasuisse-deploy:type=Connector,name="connector.file.mule.default.j"	org.mule.module.management.mbean.ConnectorService	Information on the management interface of the MBean	Unregister
Mule.mule.rts.betasuisse-deploy:type=Endpoint,service="mule.rts.xml-betabab-MainProcess",connector=connector.file.mule.default,name="endpoint.file.data.dev.metasem.betasuisse.fto.bab.in"	org.mule.module.management.mbean.EndpointService	Information on the management interface of the MBean	Unregister
Mule.mule.rts.betasuisse-deploy:type=Flow,name="mule.rts.xml-betabab-MainProcess"	org.mule.module.management.mbean.FlowConstructService	Information on the management interface of the MBean	Unregister
Mule.mule.rts.betasuisse-deploy:type=Model,name=" muleSystemModel(seda)"	org.mule.module.management.mbean.ModelService	Information on the management interface of the MBean	Unregister
Mule.mule.rts.betasuisse-deploy:type=org.mule.Notification,name=MuleNotificationBroadCaster	org.mule.module.management.agent.JmxServerNotificationAgent\$BroadcastNotificationService	Information on the management interface of the MBean	Unregister
Mule.mule.rts.betasuisse-deploy:type=org.mule.Notification,name=MuleNotificationListener	org.mule.module.management.agent.JmxServerNotificationAgent\$NotificationListener	Information on the management interface of the MBean	Unregister
Mule.mule.rts.betasuisse-deploy:type=org.mule.Statistics,Application="application_totals"	org.mule.module.management.mbean.FlowConstructStats	Information on the management interface of the MBean	Unregister
Mule.mule.rts.betasuisse-deploy:type=org.mule.Statistics,Flow="mule.rts.xml-betabab-MainProcess"	org.mule.module.management.mbean.FlowConstructStats	Information on the management interface of the MBean	Unregister
Mule.mule.rts.betasuisse-deploy:type=Statistics,name=AllStatistics	org.mule.module.management.mbean.StatisticsService	Information on the management interface of the MBean	Unregister
Domain: Mule.mule.rts.management-console			
Mule.mule.rts.management-console:name=Configuration	org.mule.module.management.mbean.MuleConfigurationService	Information on the management interface of the MBean	Unregister
Mule.mule.rts.management-console:name=MuleContext	org.mule.module.management.mbean.MuleService	Information on the management interface of the MBean	Unregister
Mule.mule.rts.management-console:name=Mx4jHttpAdapter	mx4j.tools.adaptor.http.HttpAdaptor	Information on the management interface of the MBean	Unregister
Mule.mule.rts.management-console:type=Application,name="application_totals"	org.mule.module.management.mbean.ApplicationService	Information on the management interface of the MBean	Unregister
Mule.mule.rts.management-console:type=Model,name=" muleSystemModel(seda)"	org.mule.module.management.mbean.ModelService	Information on the management interface of the MBean	Unregister
Mule.mule.rts.management-console:type=org.mule.Notification,name=MuleNotificationBroadCaster	org.mule.module.management.agent.JmxServerNotificationAgent\$BroadcastNotificationService	Information on the management interface of the MBean	Unregister
Mule.mule.rts.management-console:type=org.mule.Notification,name=MuleNotificationListener	org.mule.module.management.agent.JmxServerNotificationAgent\$NotificationListener	Information on the management interface of the MBean	Unregister
Mule.mule.rts.management-console:type=org.mule.Statistics,Application="application_totals"	org.mule.module.management.mbean.FlowConstructStats	Information on the management interface of the MBean	Unregister
Mule.mule.rts.management-console:type=Statistics,name=AllStatistics	org.mule.module.management.mbean.StatisticsService	Information on the management interface of the MBean	Unregister
Domain: Mule.mule.rts.move-and-rename			

Annexe 3 : Console Management JMX

Attributes			
Name	Description	Type	Value
AsyncEventsReceived	Attribute exposed for management	long	97
AverageProcessingTime	Attribute exposed for management	long	1
ExecutionErrors	Attribute exposed for management	long	0
FatalErrors	Attribute exposed for management	long	0
MaxProcessingTime	Attribute exposed for management	long	31
MinProcessingTime	Attribute exposed for management	long	1
Name	Attribute exposed for management	java.lang.String	application totals
ProcessedEvents	Attribute exposed for management	long	97
Statistics	Attribute exposed for management	javax.management.ObjectName	Mule,mule_rts.thumbfile-requester:type=org.mule.Statistics,Application="application_totals"
SyncEventsReceived	Attribute exposed for management	long	0
TotalEventsReceived	Attribute exposed for management	long	97
TotalProcessingTime	Attribute exposed for management	long	127
Type	Attribute exposed for management	java.lang.String	Application