



Configuration Interface for MESSage ROUTing

## **Dossier de spécifications externes**

---

---

Date : 16/04/07  
Version : 1.0  
Statut : diffusable

Auteurs : **BAGNARD Natacha**  
**FOROT Julien**

## Tables des révisions

Version	Date	Modifications
0.1	14/02/07	Création du document
0.2	16/04/07	Modification du document suite à une relecture de Jérôme Camilleri
1.0	16/04/07	Validation par Jérôme Camilleri

# Table des matières

<b>1.Introduction .....</b>	<b>5</b>
1.1.Objectifs du document.....	5
1.2.Portée du document.....	5
1.3.Documentation de référence.....	5
1.4.Glossaire.....	5
<b>2.Spécifications générales.....</b>	<b>5</b>
2.1.Description des services attendus.....	5
2.1.1.Charger un graphe.....	6
2.1.2.Sauvegarder un graphe.....	7
2.1.3.Créer un graphe.....	7
2.1.4.Modifier un graphe.....	8
2.1.5.Démarrer ServiceMix.....	8
2.1.6.Arrêter ServiceMix.....	8
2.1.7.Créer un package JBI.....	9
2.1.8.Déployer un package JBI.....	9
2.1.9.Migrer un fichier de Cimero1 vers Cimero2.....	9
2.1.10.Ajouter un composant.....	10
2.1.11.Éditer un composant.....	10
2.1.12.Générer une tâche Ant pour un flux.....	11
2.1.13.Monitorer un flux.....	11
2.2.Manipulation des graphes de flux.....	12
2.2.1.Utiliser un composant.....	12
2.2.2.Supprimer un composant.....	13
2.2.3.Ajouter un lien entre 2 composants.....	13
2.2.4.Supprimer un lien entre 2 composants.....	13
2.2.5.Modifier les propriétés d'un composant.....	13
<b>3.Spécifications IHM.....</b>	<b>14</b>
3.1.Arbre des tâches.....	14
3.1.1.Légende.....	14
3.1.2.Manipuler des graphes.....	15
3.1.3.Utiliser ServiceMix.....	17
3.1.4.Manipuler les packages.....	18
3.1.5.Customiser le plugin.....	18
3.2.IHM abstraite.....	19
3.2.1.Interface globale de design de flux.....	19
3.2.2.Palette.....	20

---

3.2.3.Zone de dessin.....	21
3.2.4.Propriétés.....	21
3.2.5.Interface de configuration du plugin.....	22
3.2.6.Zone de gestion des composants.....	22
3.2.7.Interface de création et d'édition de composants.....	23
3.2.8.Zone de description des propriétés du composant.....	23
3.3.Charte graphique.....	24
3.3.1.Représentation des composants.....	24
3.3.2.Représentation des liens.....	25
<b>4.Maquettes.....</b>	<b>27</b>
4.1.1.Interface globale de design de flux.....	27
4.1.2.Interface de configuration du plugin.....	28
4.1.3.Interface de création et d'édition de composants.....	29

# **1.Introduction**

## **1.1.Objectifs du document**

Ce document présente le dossier de spécifications externes (DES) du projet CIMERO. Ce logiciel permettra de créer graphiquement des flux de messages qui pourront être déployé sur l' ESB ServiceMix.

Les spécifications externes entrent dans le cadre de l'organisation et de la gestion du développement logiciel. Elles ont pour objectif de décrire exactement ce que sera le système du point de vue de l'utilisateur.

## **1.2.Portée du document**

Ce document est destiné :

- ✓ au responsable du stage, Jérôme Camilleri
- ✓ à la consultante : Martine Tasset
- ✓ au jury du Master 2 Pro GI pour l'évaluation du stage
- ✓ à l'équipe projet : Natacha Bagnard et Julien Forot

## **1.3.Documentation de référence**

Le présent document fait référence au Cahier des Charges du projet (« CahierDesCharges.odt ») et est rédigé en fonction des clauses qualités définies dans le Plan d'Assurance Qualité Logicielle (« PlanAssuranceQualiteLogicielle.odt »).

## **1.4.Glossaire**

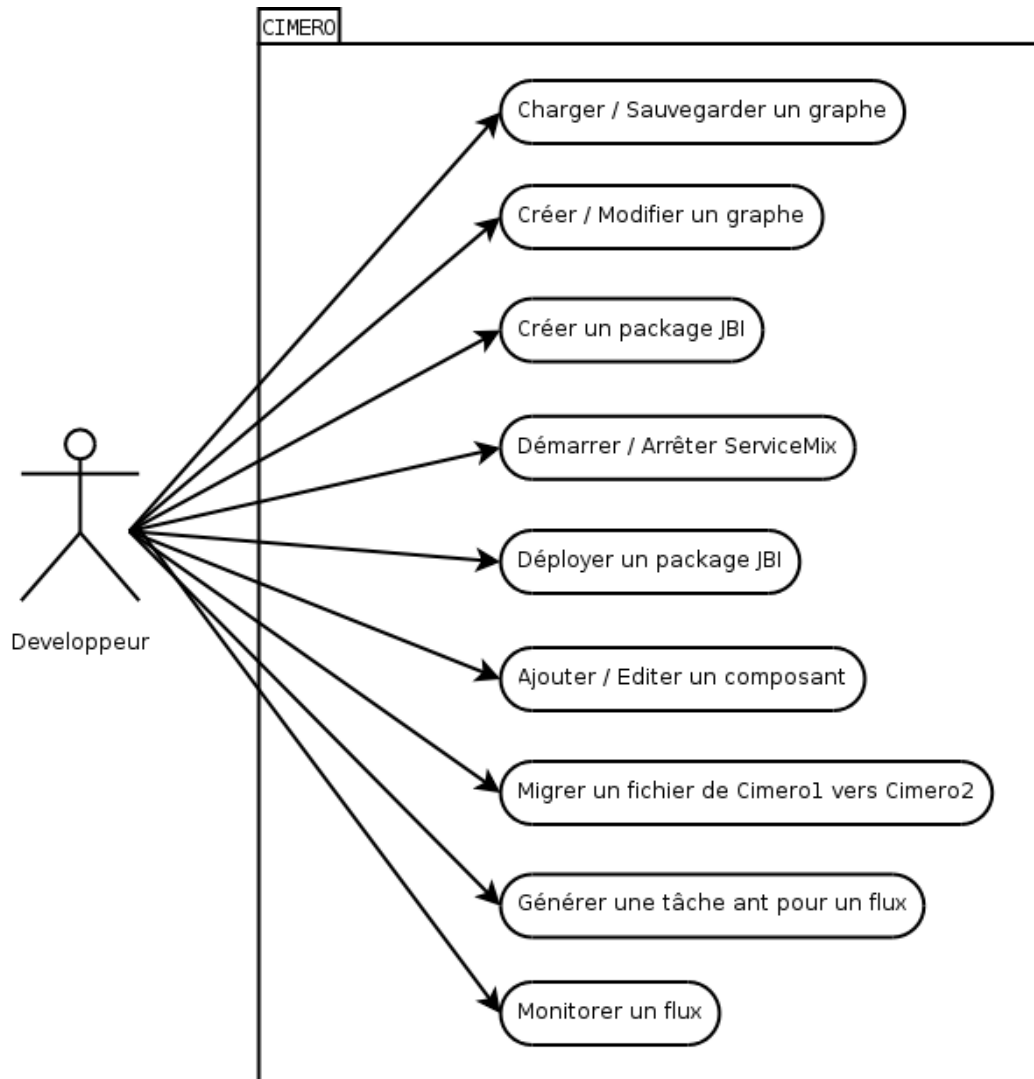
--> voir le section 1.3 (« Définitions, acronymes et abréviations ») du Cahier des Charges (CahierDesCharges.odt)

# **2.Spécifications générales**

## **2.1.Description des services attendus**

L'objet de ce logiciel est de fournir une interface graphique permettant de gérer des flux de messages entre services qui seront ensuite déployés sur les ESBs ServiceMix ou Petals grâce à des fichiers de configuration générés automatiquement.

Le graphe des cas d'utilisation suivant, présente les diverses fonctionnalités proposées par CIMERO.



*Illustration 1: graphe des cas d'utilisations de CIMERO*

### 2.1.1.Charger un graphe

- **But** : L'utilisateur peut charger le fichier contenant un graphe (fichier \*.cimero2). Celui-ci sera affiché dans la zone de dessin<sup>1</sup>.
- **Pré-condition** : Un fichier d'extension .cimero2 doit être sélectionné par l'utilisateur.
- **Exécution** : L'utilisateur ouvre le fichier.
- **Post-condition** : La représentation du graphe peut être valide, le graphe contenu dans le fichier est affiché dans la zone de dessin. La représentation du graphe peut comporter des erreurs qui empêchent l'affichage du graphe, l'utilisateur reçoit un message d'erreur

<sup>1</sup> Voir section 3.2 pour la description des différentes zones de l'interface

l'avertissant que le fichier est erroné et que le graphe ne peut être affiché.

### *2.1.2.Sauvegarder un graphe*

- **But** : L'utilisateur peut sauvegarder le graphe affiché dans la zone de dessin, celui-ci sera sauvegarder dans un fichier \*.cimero2 et pourra être chargé ultérieurement.
- **Pré-condition** : Un graphe doit être en cours d'utilisation.
- **Exécution** :
  - L'utilisateur peut choisir d'enregistrer son travail. Ceci nécessite qu'un fichier .cimero2 corresponde déjà au graphe. La représentation actuelle du graphe sera enregistrée dans le fichier.
  - L'utilisateur peut choisir d'enregistrer sous son travail, il devra préciser un nom pour le fichier .cimero2 dans lequel la représentation actuelle du graphe sera enregistrée.
- **Post-condition** : Le graphe est enregistré dans le fichier .cimero2 de nom donné. Lors d'un chargement, on doit retrouver le même graphe que lors de l'enregistrement.

### *2.1.3.Créer un graphe*

- **But** : L'utilisateur peut créer un nouveau fichier .cimero2 ne contenant aucune donnée de flux. La zone de dessin correspondant à l'affichage du graphe de ce fichier est également vierge.
- **Pré-condition** : Aucune.
- **Exécution** : L'utilisateur choisit de créer un nouveau fichier de type cimero. Il doit spécifier un nom et l'ESB qu'il souhaite utiliser pour déployer ensuite les services correspondant au flux qu'il va créer. Si le nom choisi n'existe pas, alors le fichier est créé. Sinon, un message d'erreur l'avertit que celui-ci est déjà affecté à un autre graphe et qu'il doit choisir un autre nom.
- **Post-condition** : Le fichier créé est accessible dans le projet auquel il a été ajouté. Il ne contient aucune donnée concernant un quelconque flux. La palette associée au graphe contient uniquement les composants utilisables avec l'ESB préalablement choisi.

#### *2.1.4.Modifier un graphe*

- **But** : L'utilisateur peut modifier un graphe.
- **Pré-condition** : Un fichier .cimero2 doit être chargé.
- **Exécution** : L'utilisateur peut ajouter ou supprimer des composants, ajouter ou supprimer des liens entre les différents composants du graphe ( tant que ceux-ci sont valides ) ou encore modifier les propriétés des composants<sup>1</sup>.
- **Post-condition** : La vue de la zone de dessin est mise à jour automatiquement.

#### *2.1.5.Démarrer ServiceMix*

- **But** : L'utilisateur peut lancer ServiceMix depuis l'environnement de travail du plugin.
- **Pré-condition** : Aucune.
- **Exécution** : L'utilisateur choisit de démarrer ServiceMix. Les informations qu'il renvoi sont accessibles depuis l'environnement du plugin. Soit le démarrage s'est déroulé sans problème et un message indique qu'il est prêt à recevoir des services à déployer, soit il a rencontré un problème et un message d'erreur est affiché.
- **Post-condition** : Si le serveur a correctement démarré, les services d'un flux peuvent y être déployé depuis l'environnement du plugin.

#### *2.1.6.Arrêter ServiceMix*

- **But** : L'utilisateur peut arrêter ServiceMix depuis l'environnement de travail du plugin.
- **Pré-condition** : Un serveur ServiceMix doit être démarré.
- **Exécution** : L'utilisateur choisit de stopper le serveur démarré. Les informations qu'il renvoi sont accessible depuis l'environnement du plugin. Soit le serveur s'arrête normalement et un message l'indique à l'utilisateur, soit il rencontre des problèmes et les erreurs sont affichés.
- **Post-condition** : Le serveur est arrêté, les services déployés sur celui-ci ne sont plus accessibles.

---

<sup>1</sup> Ces différentes actions de modification seront explicités plus en détail dans la section \*.\*.



### *2.1.7. Créer un package JBI*

- **But** : L'utilisateur peut créer une archive .zip contenant tous les fichiers de configuration concernant son flux. Celui-ci peut ensuite être déployée.
- **Pré-condition** : Un fichier .cimero2 doit être chargé et validé.
- **Exécution** : L'utilisateur choisit de générer le package JBI du graphe qu'il a chargé. Il doit préalablement l'avoir validé pour s'assurer que celui-ci ne contient pas d'incohérence qui pourrait provoquer des erreurs lors du déploiement. Si une erreur survient au moment de la génération du package (si par exemple la validation n'a pas été effectuée et que des incohérences subsistent dans le graphe), un message d'erreur est transmis à l'utilisateur lui indiquant que le package n'a pas pu être généré.
- **Post-condition** : Le package JBI généré est spécifique à un ESB (celui choisi lors de la création du fichier .cimero2). Il permet de déployer les services et leur chaînage représenté sur le graphe et d'utiliser ce flux. L'arborescence correspondant à ce package est conservée, elle apparaît avec l'archive dans le projet cimero. Le répertoire principal de l'arborescence et l'archive ont le même nom que le fichier .cimero2 correspondant.

### *2.1.8. Déployer un package JBI*

- **But** : L'utilisateur peut déployer les services d'un flux sur l'ESB correspondant à celui associé au graphe. Ce flux est ensuite accessible et utilisable.
- **Pré-condition** : Un graphe est chargé et le package JBI correspondant a été généré sans erreur. L'ESB associé au graphe est lancé.
- **Exécution** : L'utilisateur choisit de déployer le flux correspondant au graphe. Aucune information supplémentaire ne lui est demandée. Le serveur sur lequel le flux est déployé affiche les informations concernant le déroulement de ce déploiement. Si il n'y a pas de problème, chaque SU et SA sont déployés sans message d'erreur, sinon un message d'erreur est affiché par le serveur.
- **Post-condition** : Le flux déployé est utilisable. Il permet l'envoi de messages aux services d'entrée, routage des messages et réception par les services en sortie.

### *2.1.9. Migrer un fichier de Cimero1 vers Cimero2*

- **But** : L'utilisateur peut choisir, dans Cimero1, un fichier pour lequel il souhaite générer un fichier XML intermédiaire associé puis l'importer dans Cimero2 et visualiser le graphe correspondant.

- **Pré-condition** : Un fichier .cimero existe.
- **Exécution** : L'utilisateur décide de migrer un graphe de Cimero1 vers Cimero2. Il choisit un fichier XML intermédiaire généré avec Cimero1 contenant les informations du graphe qu'il souhaite importer dans Cimero2. Soit la génération s'effectue sans erreurs et un fichier .cimero2 correspondant est créé, le graphe correspondant est affiché dans la zone de dessin et la palette correspondant à l'ESB est chargée. Si le package est correct mais utilise des composants non supportés par la nouvelle version de CIMERO, l'importation sera effectuée en mode dégradée. Un icône « composant non reconnu » remplacera le composant dans le graphe. Les propriétés de ce composant ne seront pas accessibles. L'utilisateur pourra choisir de remplacer ce composant par un composant supporté par CIMERO ou utiliser la fonction d'Ajout d'un composant. Si une erreur survient pendant la génération, l'utilisateur reçoit un message d'erreur et aucun fichier .cimero2 n'est généré.
- **Post-condition** : Le fichier .cimero2 généré porte le même nom que le fichier XML intermédiaire auquel il correspond. Il apparaît dans le projet et peut-être modifié.

#### *2.1.10. Ajouter un composant*

- **But** : L'utilisateur peut ajouter un nouveau composant qui sera ensuite utilisable dans les flux créés.
- **Pré-condition** : Le composant doit respecter les standards JBI<sup>1</sup>.
- **Exécution** : L'utilisateur doit préciser les propriétés de son composant et l'ESB avec lequel il est compatible. Chacune des propriétés du composant devra être paramétrable pour les composant de ce type dans un graphe.
- **Post-condition** : Le composant ajouté est accessible dans la palette correspondant à l'ESB compatible (Il peut être compatible avec les 2). Il peut être utilisé et configuré dans un graphe.

#### *2.1.11. Éditer un composant*

- **But** : L'utilisateur peut Éditer un composant disponible dans la palette.
- **Pré-condition** : Le composant doit respecter les standards JBI et doit déjà être présent dans la palette.

---

<sup>1</sup> Voir la spécification : Java Business Integration 1.0 – Final Release

- **Exécution** : L'utilisateur doit préciser les propriétés de son composant et l'ESB avec lequel il est compatible. Chacune des propriétés du composant devra être paramétrable pour les composant de ce type dans un graphe. La compatibilité avec les graphes existant utilisant ce composant ne sera pas garantie.
- **Post-condition** : La modification du composant est enregistrée. Il peut utilisé et configuré dans un graphe.

#### *2.1.12.Générer une tâche Ant pour un flux*

- **But** : L'utilisateur peut générer un fichier contenant une tâche Ant permettant de créer plusieurs packages JBI au lieu d'un seul pour permettre plus de souplesse au niveau du déploiement (Les services peuvent ainsi être déployés sur des machines différentes).
- **Pré-condition** : Un fichier .cimero2 doit être chargé et validé.
- **Exécution** : L'utilisateur choisi de générer une tâche Ant pour le graphe qu'il a chargé. Il doit préalablement l'avoir validé pour s'assurer que celui-ci ne contient pas d'incohérence qui pourrait provoquer des erreurs lors du déploiement. Si une erreur survient au moment de la génération du fichier, un message d'erreur est transmis à l'utilisateur lui indiquant que le fichier n'a pas pu être généré.
- **Post-condition** : Le fichier xml contenant la tâche Ant apparaît dans le projet. De plus, si l'arborescence des répertoires correspondant aux SAs et SUs n'existe pas, elle est créée en même temps.

#### *2.1.13.Monitorer un flux*

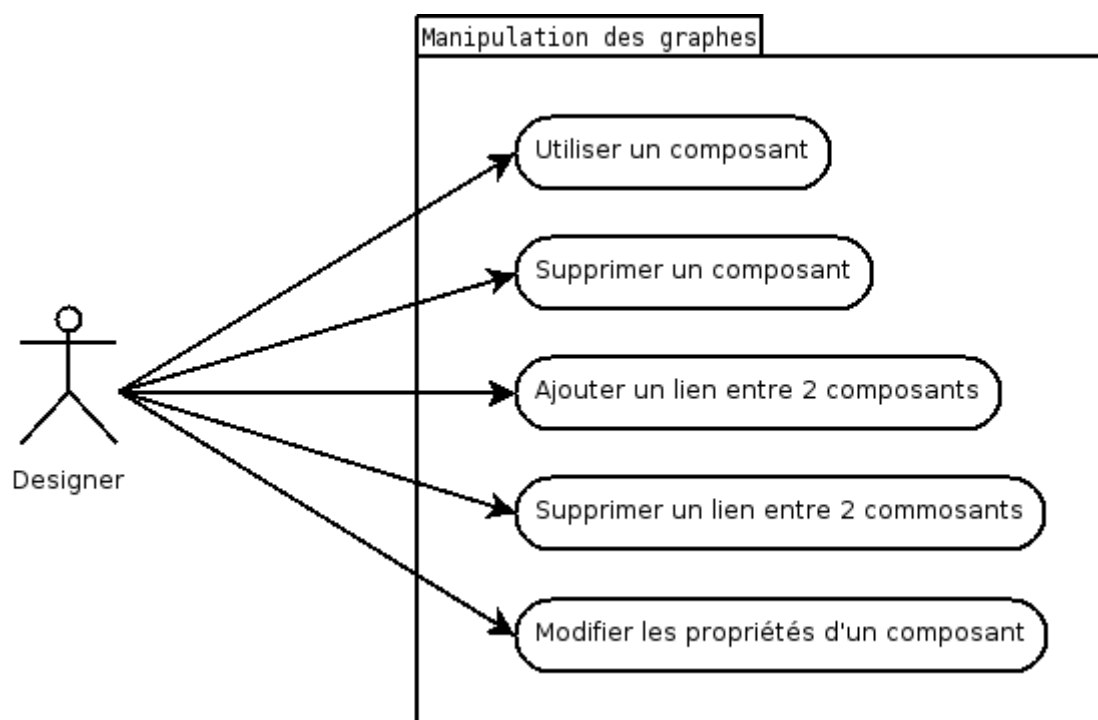
- **But** : L'utilisateur peut tester le flux qui sera généré à partir du graphe. Lors du test, il pourra visualiser l'échange de messages dans le flux et voir les composants source d'erreur.
- **Pré-condition** : Un fichier .cimero2 doit être chargé et validé. Le package JBI correspondant doit être généré.
- **Exécution** : L'utilisateur choisi de tester le flux correspondant au graphe qu'il a chargé. Il doit préalablement l'avoir validé pour s'assurer que celui-ci ne contient pas d'incohérence qui pourrait provoquer des erreurs lors du déploiement et avoir généré le package JBI correspondant. Une fois le package déployé, l'utilisateur peut suivre visuellement sur le graphe CIMERO le flux de message entre les composants de l'application JBI. Si le monitoring montre que le message s'arrête anormalement dans un composant, celui-ci

sera identifié graphiquement grâce à une couleur spécifique (rouge par exemple)

- **Post-condition** : Une fois le flux de messages terminé, le graphe initial est affiché.

## 2.2.Manipulation des graphes de flux

La manipulation d'un graphe permet à l'utilisateur d'effectuer les différentes actions suivantes :



*Illustration 2: graphe détaillé de "Manipulation des graphes" dans Cimero*

### 2.2.1.Utiliser un composant

- **But** : L'utilisateur peut ajouter dans la zone de dessin un nouveau composant disponible dans la palette.
- **Pré-condition** : Un fichier .cimero2 doit être chargé.
- **Exécution** : L'utilisateur sélectionne parmi les composants de la palette<sup>1</sup> celui qu'il souhaite ajouter à son graphe. Il précise l'endroit dans la zone de dessin où il souhaite le placer.
- **Post-condition** : Un nouveau composant est disponible dans le graphe. Ce composant est « libre », c'est-à-dire qu'il n'est lié à aucun autre.

<sup>1</sup> Voir section 3.2 pour la description des différentes zones de l'interface

### *2.2.2. Supprimer un composant*

- **But** : L'utilisateur peut supprimer un des composants disponible dans la zone de dessin.
- **Pré-condition** : Un fichier .cimero2 doit être chargé. Le graphe doit contenir au moins un composant.
- **Exécution** : L'utilisateur sélectionne un composant du graphe et le supprime.
- **Post-condition** : Le composant disparaît du graphe. Les liens entre ce composant et d'autre disparaissent. Les propriétés des composants qui étaient en relation avec celui-ci sont mises à jour si nécessaire.

### *2.2.3. Ajouter un lien entre 2 composants*

- **But** : L'utilisateur peut lier 2 composants de la zone de dessin.
- **Pré-condition** : Un fichier .cimero2 doit être chargé. Le graphe doit contenir au moins 2 composants.
- **Exécution** : L'utilisateur choisit de lier des composants. Il sélectionne le premier composant, qui sera celui à l'origine du lien, puis le second qui sera à l'arrivée. Un message d'erreur est transmis à l'utilisateur si le lien est impossible.
- **Post-condition** : Les 2 composants sont reliés par une flèche précisant le sens de l'échange.

### *2.2.4. Supprimer un lien entre 2 composants*

- **But** : L'utilisateur peut délier 2 composants.
- **Pré-condition** : Un fichier .cimero2 doit être chargé. Le graphe doit contenir au moins 2 composants liés entre eux.
- **Exécution** : L'utilisateur choisit un lien entre 2 composants du graphe et le supprime.
- **Post-condition** : La flèche qui liait les 2 composants disparaît. Les composants sont mis à jour si nécessaire.

### *2.2.5. Modifier les propriétés d'un composant*

- **But** : L'utilisateur peut spécifier des paramétrages pour chaque composant.

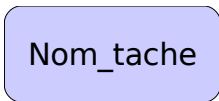








- **Pré-condition** : Un fichier .cimero2 doit être chargé. Le graphe doit contenir au moins un composant.
- **Exécution** : L'utilisateur choisit le composant qu'il souhaite paramétrer et accède à la zone des propriétés<sup>1</sup>. Il fournit les modifications qu'il souhaite appliquer aux propriétés du composant.
- **Post-condition** : Les modifications du composant sont prises en compte et la zone de dessin est mise à jour si nécessaire.

## 3. Spécifications IHM

### 3.1. Arbre des tâches

Les fonctionnalités détaillées dans la partie précédente peuvent être regroupées au sein d'un arbre de tâches, lui même décomposé en sous arbres.

#### 3.1.1. Légende

	Tâche utilisateur		Entrelacement
	Tâche utilisateur correspondant à un use case		Répétition
	Tâche détaillée en sous-tâches		Ou exclusif
	Tâche abstraite		Séquence
			Optionnel

L'arbre suivant montre l'enchaînement des actions utilisateur à effectuer lors de l'utilisation des différentes fonctionnalités proposées par CIMERO.

<sup>1</sup> Voir section \*.\* pour la description des différentes zones de l'interface

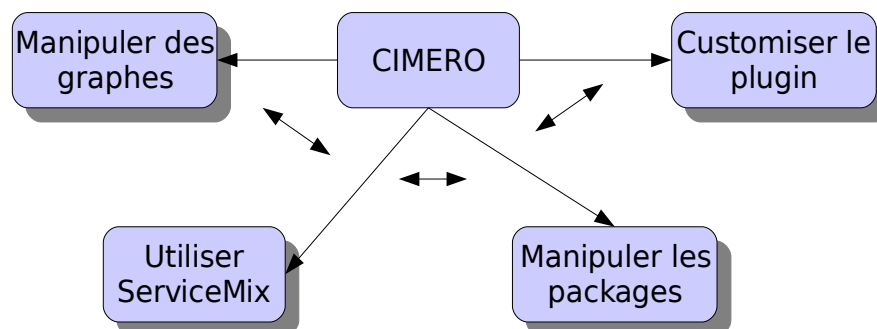


Illustration 3: Arbre des tâches du plugin CIMERO

### 3.1.2.Manipuler des graphes

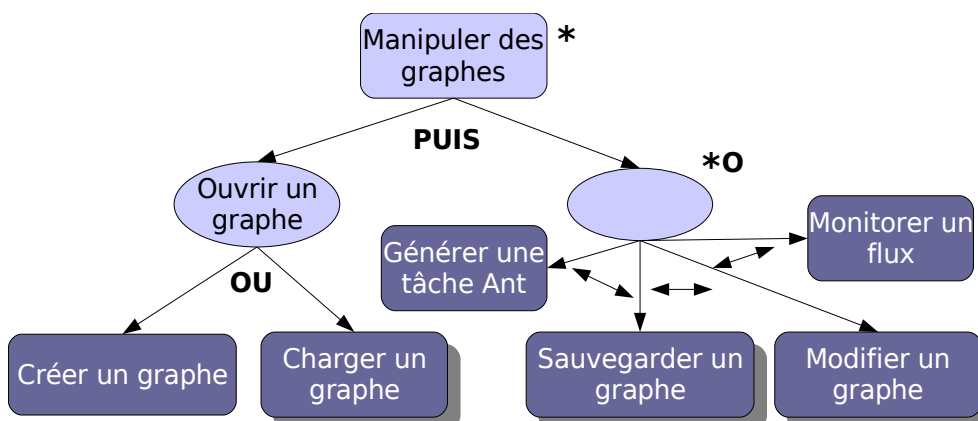
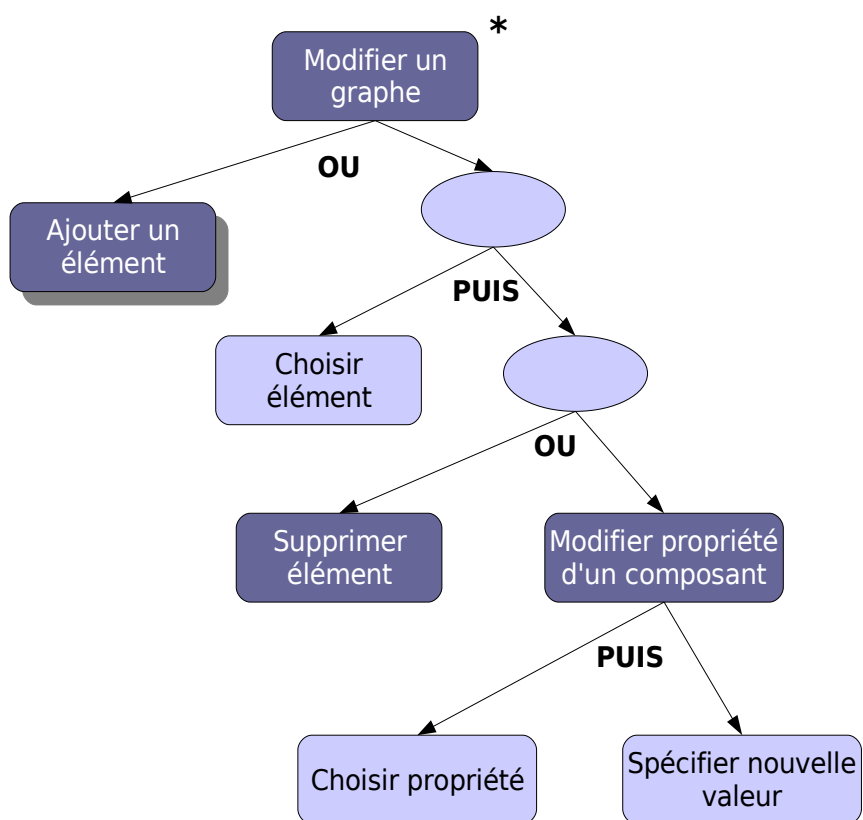


Illustration 4: Sous-arbre des tâches pour la manipulation d'un graphe

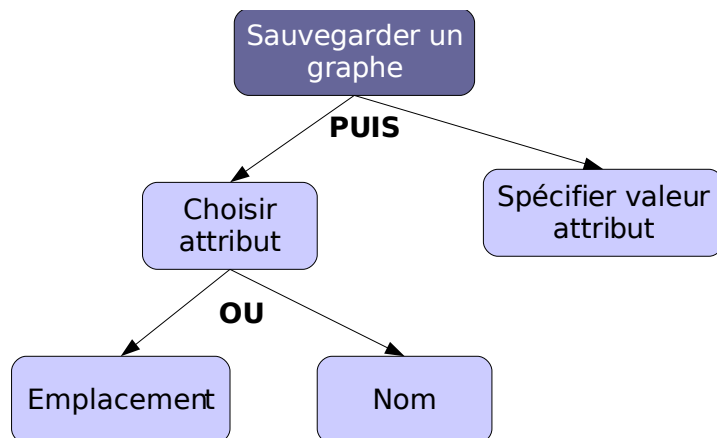
- **Modifier un graphe**



*Illustration 5: Sous-arbre des tâches pour la modification d'un graphe*

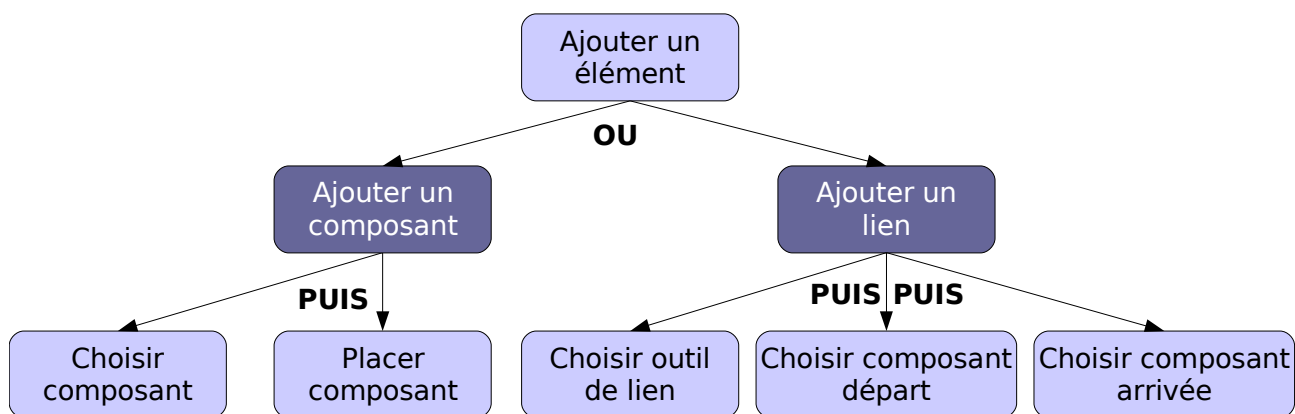


### • Sauvegarder un graphe



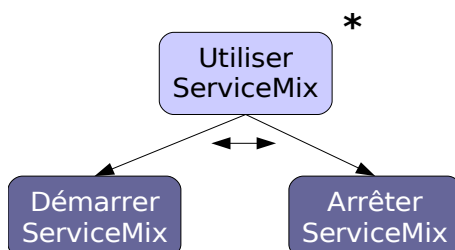
*Illustration 6: Sous-arbre des tâches pour la sauvegarde d'un graphe*

### • Ajouter un élément



*Illustration 7: Sous-arbre des tâches pour l'ajout d'un élément*

### 3.1.3.Utiliser ServiceMix



*Illustration 8: Sous-arbre des tâches pour l'utilisation de ServiceMix*

### 3.1.4.Manipuler les packages

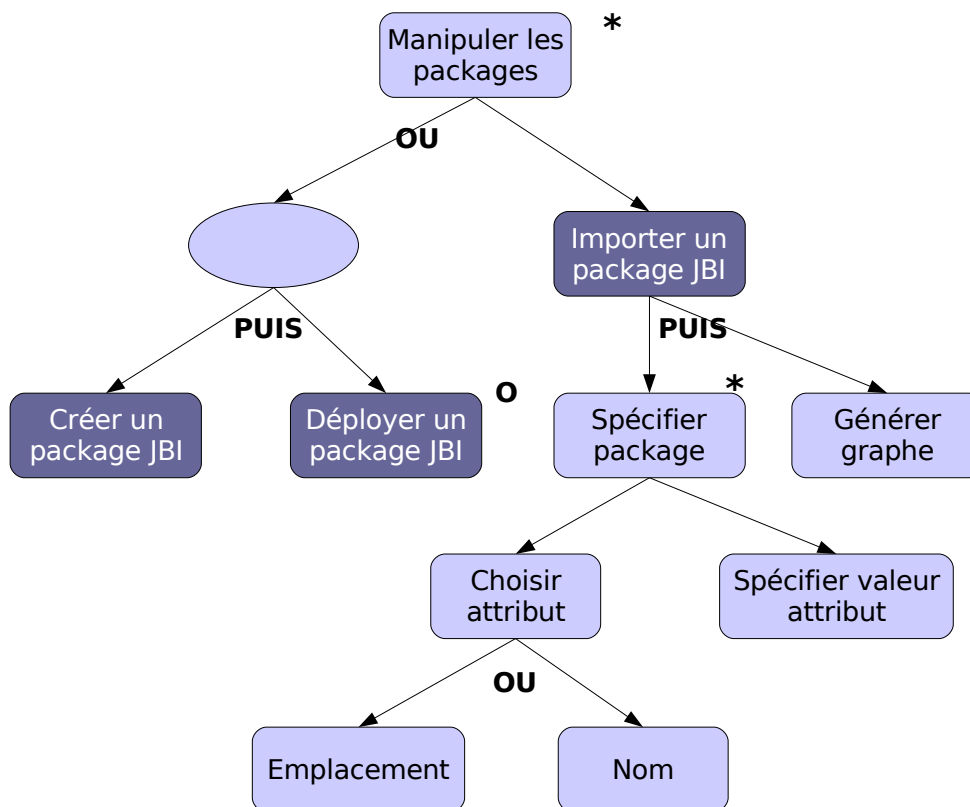


Illustration 9: Sous-arbre des tâches pour la manipulation de packages

### 3.1.5.Customiser le plugin

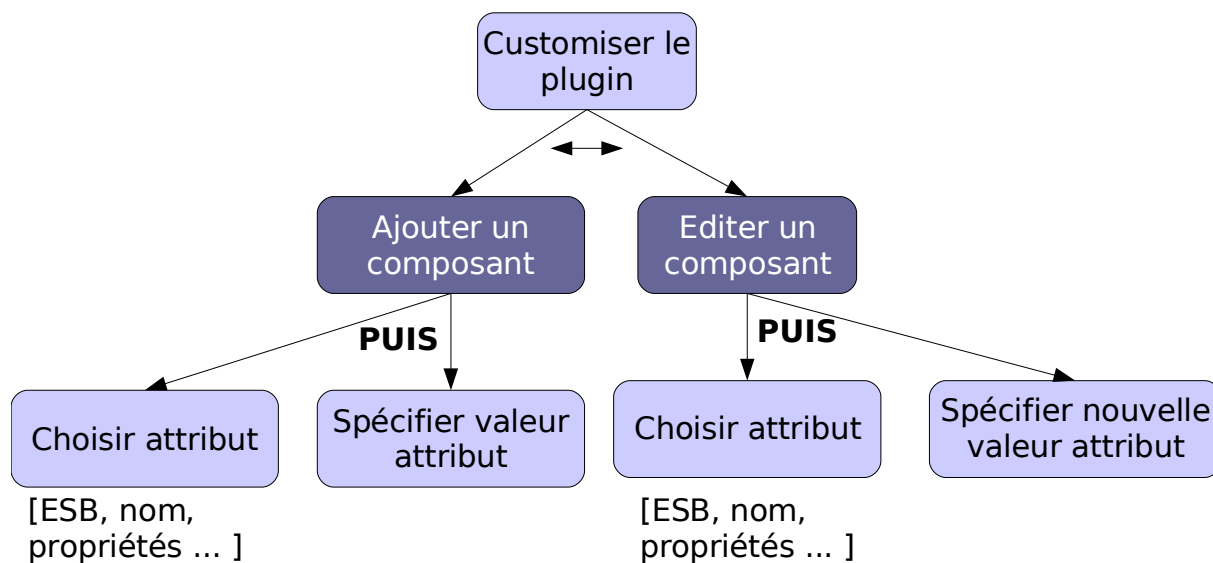
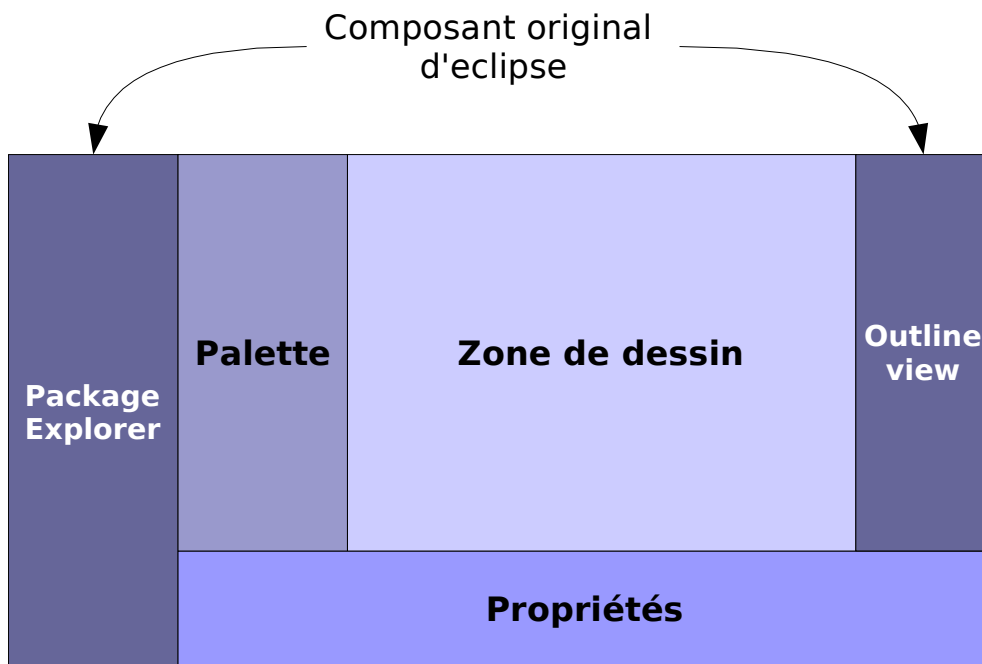


Illustration 10: Sous-arbre des tâches pour la customisation du plugin CIMERO

## 3.2.IHM abstraite

### 3.2.1.Interface globale de design de flux

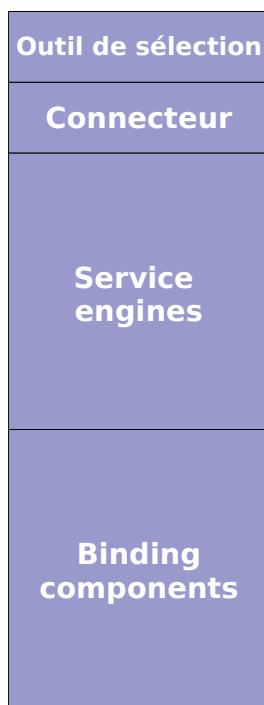
L'interface de design de flux du plugin CIMERO reste inchangée par rapport à la précédente version. Elle est composée de 3 parties principales : la palette, la zone de dessin et la zone de spécification des propriétés. Cette interface s'intègre dans celle de l'IDE eclipse.



*Illustration 11: Interface principale de cimero*

### 3.2.2. Palette

La palette contient les différents éléments dont l'utilisateur peut se servir pour construire ses flux : les composants et le connecteur. La palette est composée de plusieurs zones dynamiques. Le schéma suivant représente l'interface abstraite de la palette.



*Illustration 12: IHM  
abstraite de la palette*

La zone outil de sélection permet de se mettre en mode sélection, afin de pouvoir sélectionner les éléments du flux présents dans la zone de dessin.

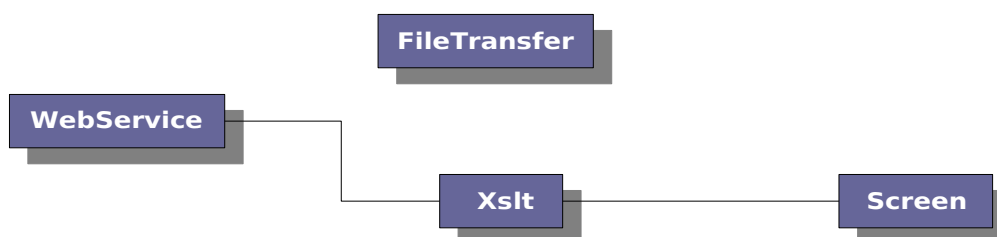
La zone connecteur contient l'outil permettant de créer des liens entre les différents composants du graphe de la zone de dessin.

Les zones Service engines et Binding components contiennent les composants utilisables dans les graphes de flux.

Les zones Service engines et Binding components sont dynamiques, elle contiennent les composants proposés avec le plugin ainsi que les différents composants qui ont été rajoutés par l'utilisateur. Les différentes zones peuvent être enroulées ou déroulées (sauf la zone outil de sélection).

### 3.2.3.Zone de dessin

La zone de dessin permet de visualiser le flux composé d'éléments liés entre eux. L'exemple suivant représente un flux en construction, visible dans la zone de dessin.



*Illustration 13: Flux en construction*

A cet instant, le flux est composé d'un lien entre un web service et une sortie à l'écran avec une transformation XSLT qui sera opérée entre le message d'entrée et celui de sortie. Un composant FileTransfer a été déposé mais n'est pas encore utilisé, il faut pour cela rajouter d'autres composants d'autres liens. Il n'y a pas de limite au nombre de composants du flux. Les liens entre composants doivent être valides pour apparaître dans la zone de dessin, on ne peut pas ajouter de lien entre le composant XSLT et le composant FileTransfer par exemple. Les composants peuvent être déplacés à volonté dans la zone de dessin, même une fois liés.

### 3.2.4.Propriétés

La zone de propriétés propose un espace de paramétrage des composants. Elle permet de voir et de modifier les propriétés d'un composant de la zone de dessin sélectionné. Chaque composant a ses propriétés spécifiques. Le schéma suivant montre la manière dont sont présentés les propriétés des composants.

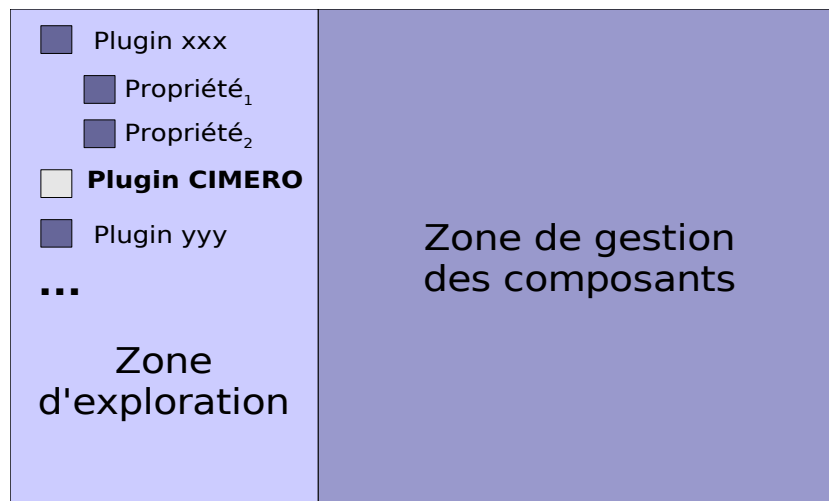
Nom <sub>1</sub>	Valeur <sub>1</sub>
Nom <sub>2</sub>	Valeur <sub>2</sub>
...	...
Nom <sub>n</sub>	Valeur <sub>n</sub>

*Illustration 14: IHM abstraite de la zone des propriétés*

Chaque propriété d'un composant est listée dans cette vue et sa valeur peut-être modifiée. Si les valeurs possibles d'une propriété sont restreintes, une liste proposant les valeurs possibles sera proposée à l'utilisateur. Une propriété n'accepte que des valeurs dont le type est valide.

### **3.2.5.Interface de configuration du plugin**

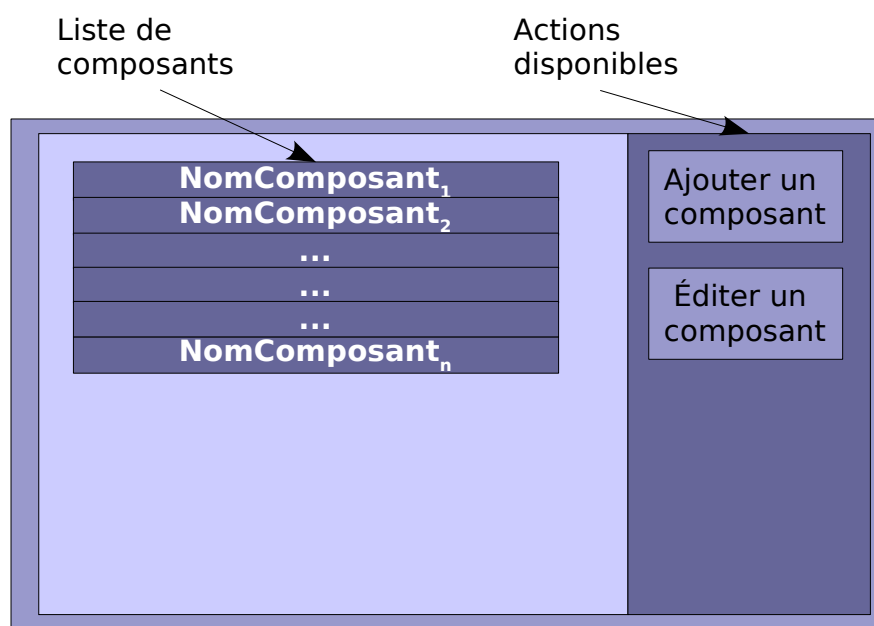
L'interface de configuration du plugin CIMERO permet d'ajouter de nouveaux composants ou de modifier le paramétrage de composants déjà installés. Elle est composée de 2 parties : Une zone d'exploration et une zone de paramétrage. Cette interface est celle proposée par Eclipse pour la gestion des préférences .



*Illustration 15: Interface de configuration du plugin*

### **3.2.6.Zone de gestion des composants**

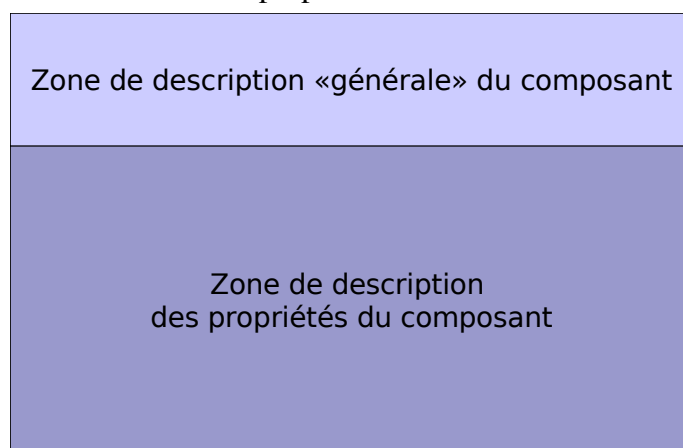
La zone de gestion des composants permet d'ajouter de nouveaux composants ou de modifier les données concernant un composant déjà disponible.



*Illustration 16: IHM abstraite de la zone de gestion des composants*

### 3.2.7. Interface de création et d'édition de composants

L'interface d'ajout et d'édition de composants permet de créer ou d'éditer un nouveau composant. La zone de description générale permet de préciser l'ESB cible, le nom du composant, les images qui lui seront associées (dans la palette, dans le graphe ...), l'adresse du composant à installer et le nombre maximal d'entrées et de sorties. La zone de description des propriétés du composant permet de préciser ses diverses propriétés.

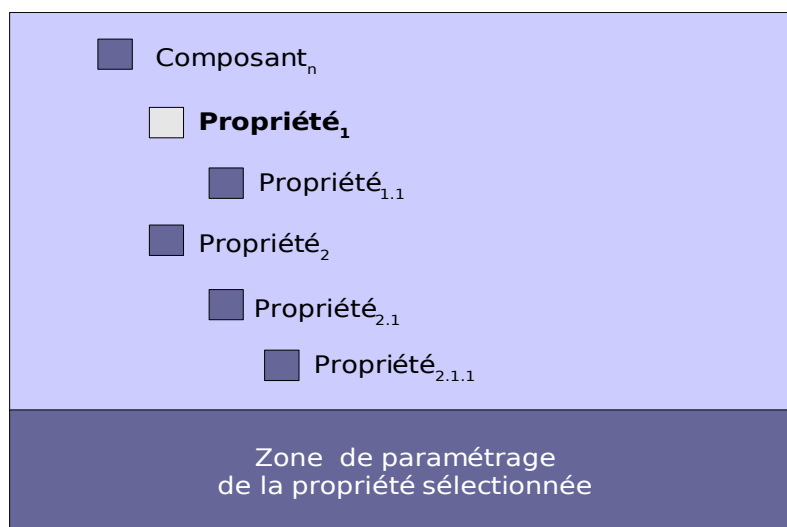


*Illustration 17: IHM abstraite pour l'ajout et l'édition de nouveaux composants*

### 3.2.8. Zone de description des propriétés du composant

La zone de description des propriétés du composant permet de visualiser ses propriétés sous forme d'arbre.

Lors de l'édition d'un composant, cette zone contiendra les différentes propriétés déjà affectées à celui-ci. Lors de l'ajout d'un nouveau composant, elle ne contient qu'un noeud racine «component». Il est ensuite possible d'ajouter ou supprimer des fils à chaque noeud.



*Illustration 18: IHM abstraite de la zone de description des composants*

La zone de paramétrage des composants se présente comme la zone propriété de la section 2.4 avec pour chacun des paramètres de la propriétés une valeur ou une liste de valeurs.


Les paramètres d'une propriété sont son nom, sa description, si la propriété est spécifique à l'utilisation en tant que consumer, provider ou aucun des 2, si elle doit être obligatoirement définie lors de l'instanciation, si les valeurs possibles sont contraintes (liste des valeur possibles ou pattern) et le type.

### 3.3.Charte graphique

La zone de dessin et la palette respecte la charte graphique suivante afin de faciliter l'utilisabilité du plugin CIMERO.

#### 3.3.1.Représentation des composants

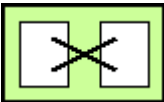
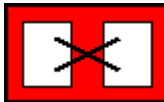



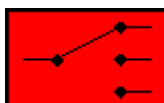
Pour chaque composant, 3 représentations doivent être proposées :

- Une de base disponible dans la palette et dans le graphe si il n'y a pas d'erreur.
- Une pour la mise en valeur des erreurs relevées lors de la validation. C'est la même que celle de base mais avec l'icône  dans le coin haut droit.
- Une autre pour la mise en valeur d'une erreur dans le graphe, lors du test du flux avec le monitoring.

#### • Service Engines

Les représentations des « Service Engines » proposées par défaut par CIMERO<sup>1</sup> sont issues si possible du livre Enterprise Integration Pattern<sup>2</sup>, sinon une représentation proche de celle proposé dans le livre sera créée.

Les représentations suivantes sont celles des composants proposés par défaut par CIMERO.

Nom du composant	Basique	Erreur
servicemix-saxon		
servicemix-drools		
servicemix-eip:content-based router		

<sup>1</sup> Voir section 3.3.3 de CahierDesCharges.odt

<sup>2</sup> Voir <http://www.enterpriseintegrationpatterns.com/toc.html>



servicemix-eip:wire-tap		
servicemix-eip:content-enricher		
servicemix-eip:split-aggregator		
servicemix-eip:xpath-splitter		
servicemix-eip:message-filter		

### • Binding Components

Les représentations des « Binding Components » proposés par défaut par CIMERO<sup>1</sup> sont composés du nom abrégé du composant (maximum 4 lettres) dans un rectangle de même taille que ceux des « service engines » mais de couleur différente : blanc . La police utilisée pour le nom est Sans Bold, Taille 24. Comme pour les « service engines ».

Les représentations suivantes sont celles des composants proposés par défaut par CIMERO.

### 3.3.2.Représentation des liens

Les liens entre 2 composants peuvent être de plusieurs type différent en fonction du MEP du composant d'arrivée.




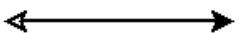
Les 4 types de MEP<sup>2</sup> sont les suivants :

- **In-Only** : Message entrant uniquement.
- **In-Out** : Message à double sens de type requête-réponse.
- **Robust In-Only** : Message entrant uniquement mais avec détection des erreurs.
- **In Optional-Out** : Message à sens unique ou à double sens en fonction de la réponse du service provider.

Une représentation différente sera associée à chaque MEP. Les représentations utilisées sont les suivantes :

<sup>1</sup> Voir section 3.3.3 de CahierDesCharges.odt

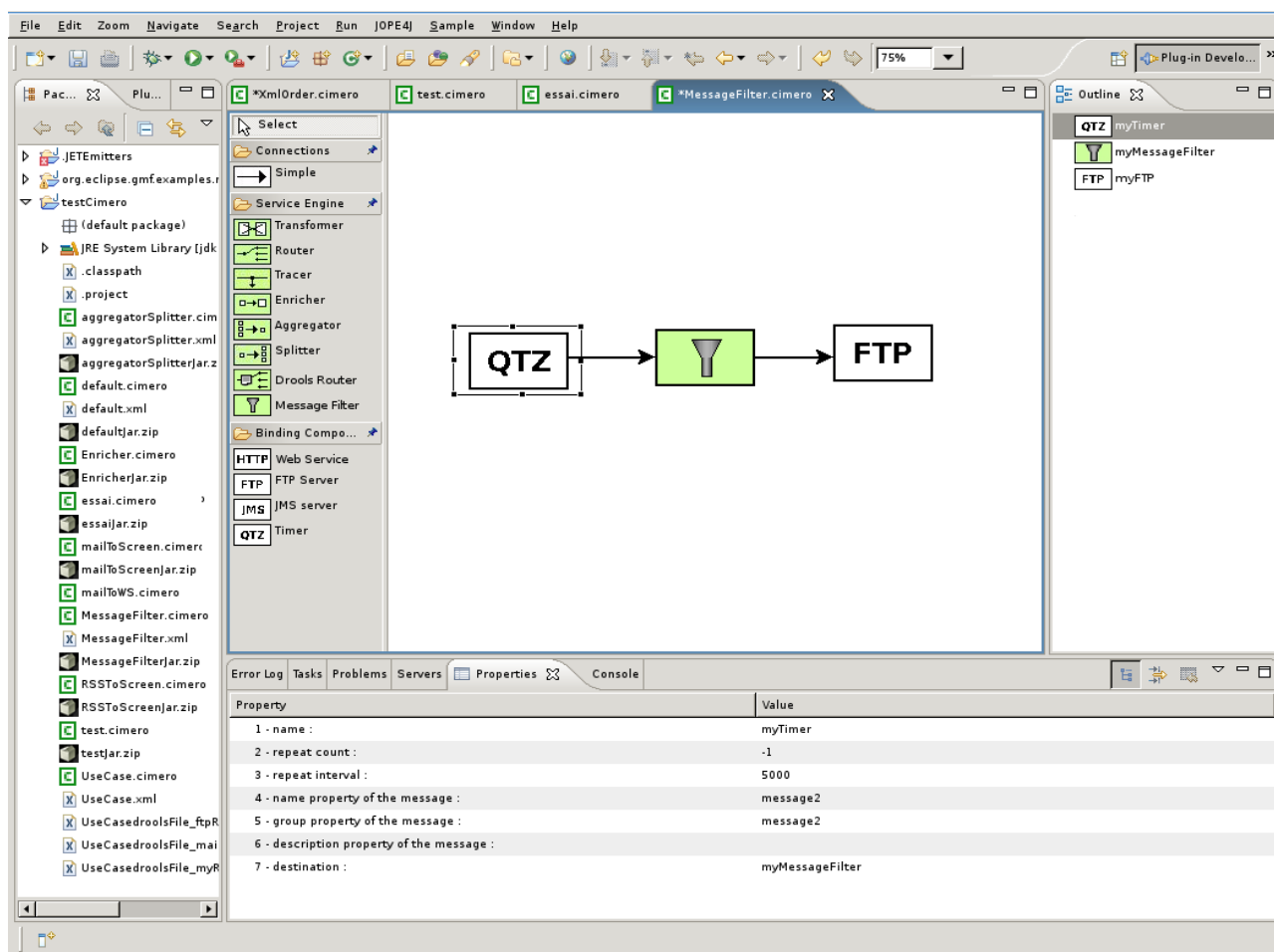
<sup>2</sup> Pour plus de précisions voir <http://java.sun.com/developer/technicalArticles/WebServices/soa3/ImplementingSOA.pdf>

<i>MEP</i>	<i>Représentation</i>
<b>In-Only</b>	
<b>In-Out</b>	
<b>Robust In-Only</b>	
<b>In Optional_Out</b>	

## 4. Maquettes

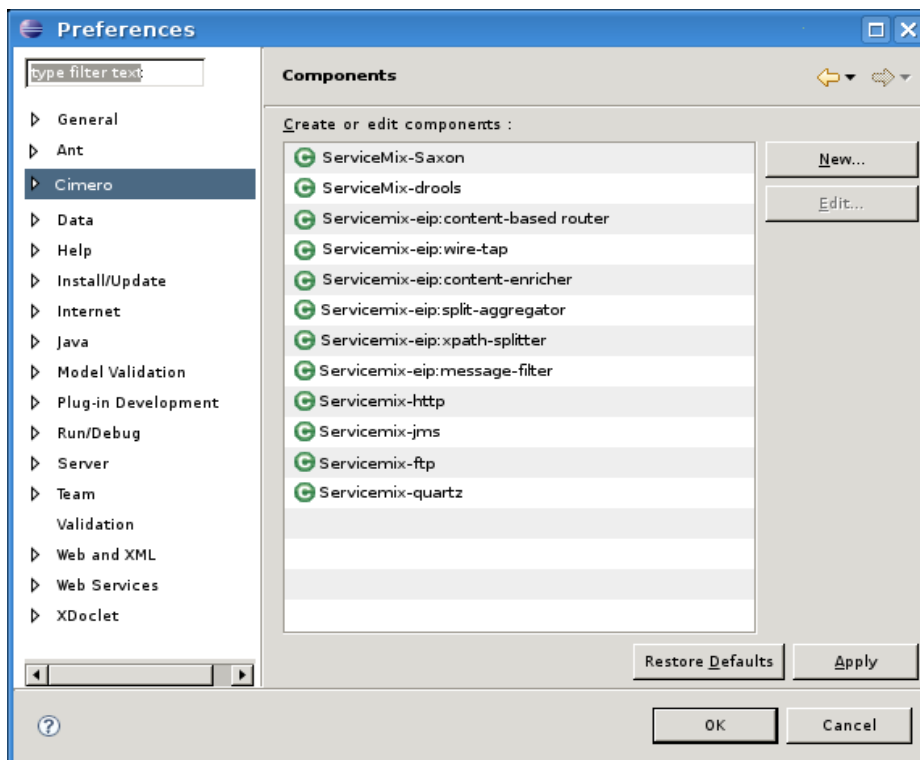
Les maquettes suivantes proposent une mise en oeuvre concrète des zones décrites précédemment pour l'IHM abstraite.

### 4.1.1. Interface globale de design de flux



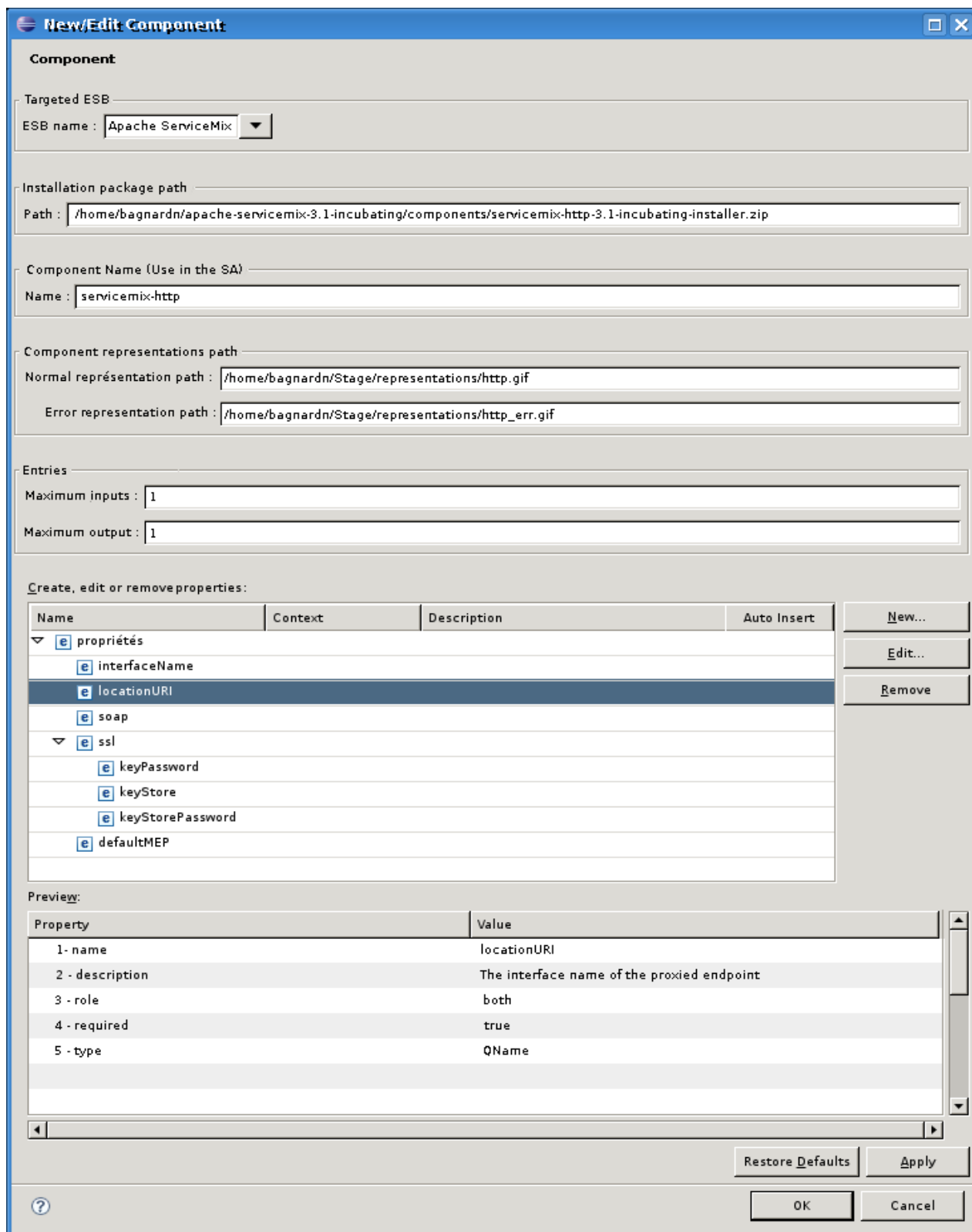
*Illustration 19: Interface principale*

### 4.1.2. Interface de configuration du plugin



*Illustration 20: Interface de configuration*

### 4.1.3. Interface de création et d'édition de composants



**Component**

Targeted ESB  
ESB name : Apache ServiceMix

Installation package path  
Path : /home/bagnardn/apache-servicemix-3.1-incubating/components/servicemix-http-3.1-incubating-installer.zip

Component Name (Use in the SA)  
Name : servicemix-http

Component representations path  
Normal representation path : /home/bagnardn/Stage/representations/http.gif  
Error representation path : /home/bagnardn/Stage/representations/http\_err.gif

Entries  
Maximum inputs : 1  
Maximum output : 1

Create, edit or remove properties:

Name	Context	Description	Auto Insert
propriétés			
interfaceName			
locationURI			
soap			
ssl			
keyPassword			
keyStore			
keyStorePassword			
defaultMEP			

Preview:

Property	Value
1 - name	locationURI
2 - description	The interface name of the proxied endpoint
3 - role	both
4 - required	true
5 - type	QName

Restore Defaults Apply OK Cancel

Illustration 21: Interface d'édition et d'ajout de composants