Obeo SmartEA - Guide Installation

v8.1.0

1. Introduction

Ce manuel est à destination des personnes susceptibles d'installer, de configurer et d'administrer SmartEA.

Lexique :

- `Obeo SmartEA Server` / `Serveur` : Le serveur d'application SmartEA.
- `Obeo SmartEA Modeler` / `Modeleur` : Client riche donnant accès aux fonctionnalités de modélisation.
- `Obeo SmartEA Module` / `Module` : Ensemble de fonctionnalités pouvant être installées dans SmartEA.
- `Obeo SmartEA Developer` / `IDE` : Une plateforme Eclipse où sont pré-installées les API SmartEA. Cet environnement permet de développer de nouveaux modules.
- `Navigateur Web` : Votre navigateur web. A utiliser pour consulter les pages web de SmartEA. L'adresse par défaut est http://localhost:8080

2. Pré-requis techniques

Vous trouverez dans cette section les pré-requis techniques d'installation.

2.1. Pré-requis matériels

Le modeleur et le serveur nécessitent la configuration matérielle minimale suivante :

- Processeur **2 Ghz 64-bit (x64)**
- **1 Go** d'espace disque

Le serveur nécessite **4 GB** de mémoire disponible au minimum. Le modeleur nécessite au minimum **2 GB** de mémoire disponible. Ces recommandations sont à réévaluer en fonction de la taille de vos modèles.

Le serveur et la base de données doivent être sur **la même machine** ou sur des machines différentes à condition que la latence réseau entre les deux soit inférieure à 1ms.

Nous recommandons un temps de latence de **moins de 5ms** entre le modeleur et le serveur. De manière générale, la stabilité du réseau et sa performance impactent la stabilité et la performance de la solution.

Il est nécessaire d'ouvrir un port sur le serveur (8080 par défaut).

2.2. Pré-requis logiciels

Les pré-requis logiciels de SmartEA sont les suivants :

- Serveur :
  - OpenJDK (Temurin – https://adoptium.net/) version 17 64 bits.
  - PostgreSQL version 13.x, 14.x ou 15.x.
  - Systèmes d'exploitation : Windows ou Linux.
- Modeleur :
  - OpenJDK (Temurin – https://adoptium.net/) version 17 64 bits.
  - Systèmes d'exploitation : Windows, Linux ou MacOS (mac OS 13 – Ventura).
  - Moteurs de rendu html :
    - Windows : Microsoft WebView2. L'utilisation du moteur de rendu par défaut du système d'exploitation Windows est fortement déconseillé pour des raisons de sécurité si celui-ci est Internet Explorer.
    - Linux : Webkit ou le moteur de rendu par défaut du système d'exploitation.
    - Mac : Moteur de rendu par défaut du système d'exploitation.
- IDE :
  - OpenJDK (Temurin – https://adoptium.net/) version 11 64 bits.
  - Systèmes d'exploitation : Windows ou Linux.
- Navigateur web :
  - Chrome
  - Safari
  - Firefox
  - Edge

2.2.1. Composant Microsoft WebView2

Pour utiliser le Modeleur sous Windows, le composant Microsoft WebView2 doit être installé ou déployé sur votre poste. Ce composant est utilisé comme moteur de rendu (Edge) des pages web par le Modeleur.

Deux installations alternatives sont possibles :

- Installation du composant sur la machine : *Evergreen Standalone Installer* ou
- Copie des éléments WebView2 dans un répertoire sur la machine cliente : *Fixed Version*.

Installation de WebView2 (Evergreen Standalone Installer) :

Le programme d'installation de WebView2 peut être téléchargé sur le site de Microsoft : https://developer.microsoft.com/en-us/microsoft-edge/webview2/#download-section

Dans la colonne du milieu, `Evergreen Standalone Installer`, choisissez le téléchargement correspondant à votre plateforme, puis suivez les instructions pour installer WebView2 sur votre machine.

En installant WebView2 sur la machine cliente le composant se mettra à jour automatiquement. De plus, moins d'espace disque est nécessaire, car WebView2 est partagé par toutes les applications WebView2 qui se trouvent sur le poste client.

Copie de WebView2 sur la machine cliente sans installation (Fixed Version) :

Une archive de WebView2 peut être téléchargée sur le site de Microsoft : https://developer.microsoft.com/en-us/microsoft-edge/webview2/#download-section

Dans la colonne de droite, `Fixed Version`, choisissez le téléchargement correspondant à votre plateforme.

Une fois l'archive récupérée, dézippez le fichier `.cab` sur votre disque avec 7-zip.

Modifiez ensuite le fichier `Obeo-SmartEA-Modeler.ini` afin d'ajouter la ligne suivante :

```
-Dorg.eclipse.swt.browser.EdgeDir=<chemin vers le répertoire contenant WebView2 dézippé>
```

En choisissant cette approche vous devez gérer les mises à jour de WebView2 vous-même, WebView2 n'étant pas automatiquement mis à jour sur le poste client.

Cette approche permet :

- d'utiliser WebView2 si le composant n'est pas installé sur votre machine et si vous n'avez pas les droits Administrateur sur la machine
- de packager un Modeleur avec le composant WebView2 embarqué pour distribuer à vos utilisateurs un Modeleur fonctionnel.

2.2.2. HTTPS

Pour des raisons de sécurité l'accès au serveur SmartEA doit obligatoirement se faire en HTTPS si cet accès ne se fait pas depuis la même machine que le serveur (localhost).

Ainsi un connecteur déployé sur le serveur peut accéder à celui-ci en HTTP. De même, un frontal (Apache, nginx) déployé sur la même machine que le serveur peut accéder au serveur SmartEA en HTTP. Dans tous les autres cas l'accès doit se faire en HTTPS.

2.3. Limite de responsabilité

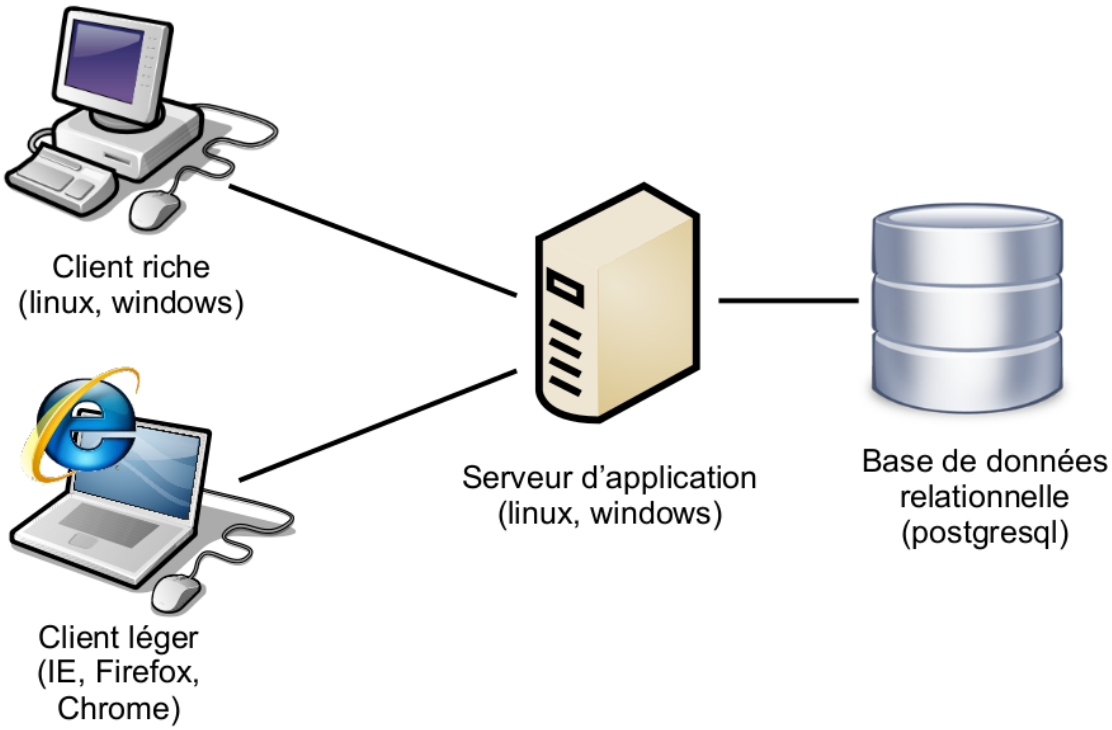
Nonobstant ce qui précède, SmartEA, Obeo Designer et Obeo Designer Team ne sont pas garantis pour fonctionner sans erreur ou interruption. Obeo n'assure aucune garantie concernant les déclarations qui sont sous le chapitre «Pré-requis techniques», ce chapitre est fourni à titre d'information.

En utilisant ces logiciels, vous reconnaissez et acceptez les risques inhérents à leur utilisation comprenant, sans limitation, l'interruption de service, la perte de connexion ou de données, le plantage du système, de mauvaises performances ou des dégradations de performance.

3. Architecture de SmartEA

SmartEA s'articule autour de trois grands composants :

- une partie serveur composée d'un serveur d'application et d'une base de données relationnelle.
- un client riche, nommé le modeleur, qui permet d'utiliser certaines fonctionnalités avancées d'interaction avec le référentiel telles que les éditeurs Sirius (diagrammes, tables, matrices)
- un client léger, consultable via un navigateur web



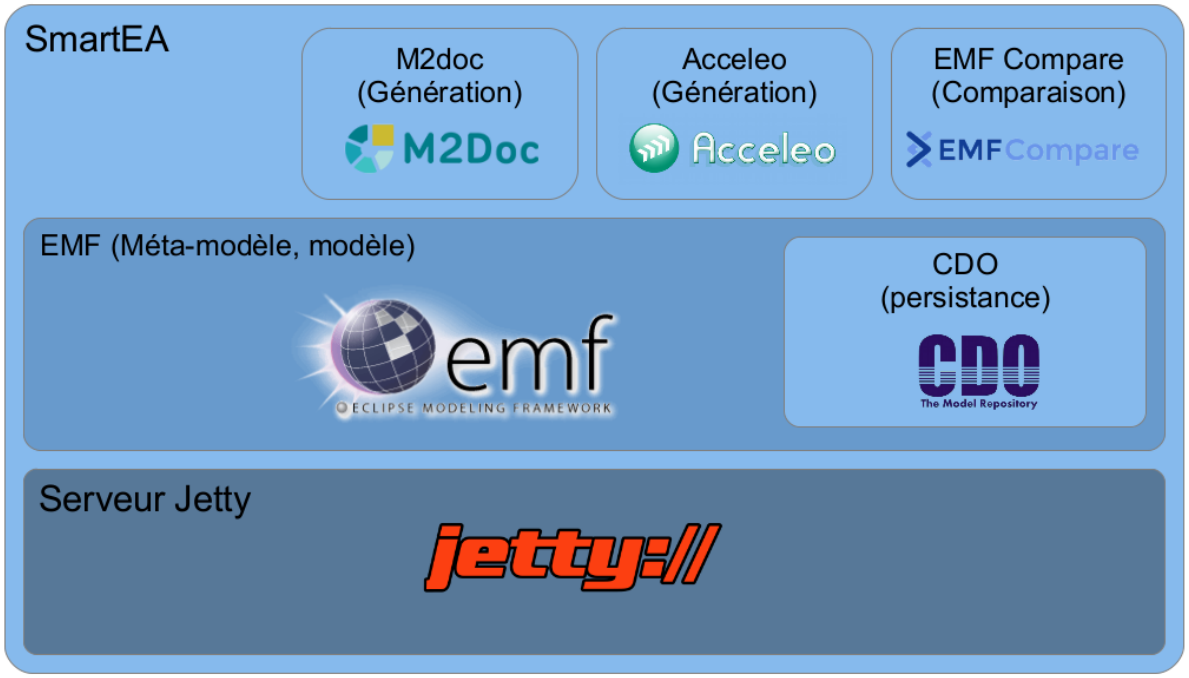
SmartEA est basé sur différentes briques logicielles, dont voici les principales :

- Obeo Designer (Viewpoint / Sirius) : http://www.obeodesigner.com/
- Acceleo : https://www.eclipse.org/acceleo/
- M2doc (MS Word documents generator) : http://www.m2doc.org/
- EMF (Eclipse Modeling Framework) : https://www.eclipse.org/emf/
- CDO (Connected Data Objects) : http://www.eclipse.org/cdo/
- Jetty (Servlet Engine and Http Server) : https://www.eclipse.org/jetty/

3.1. Architecture du serveur

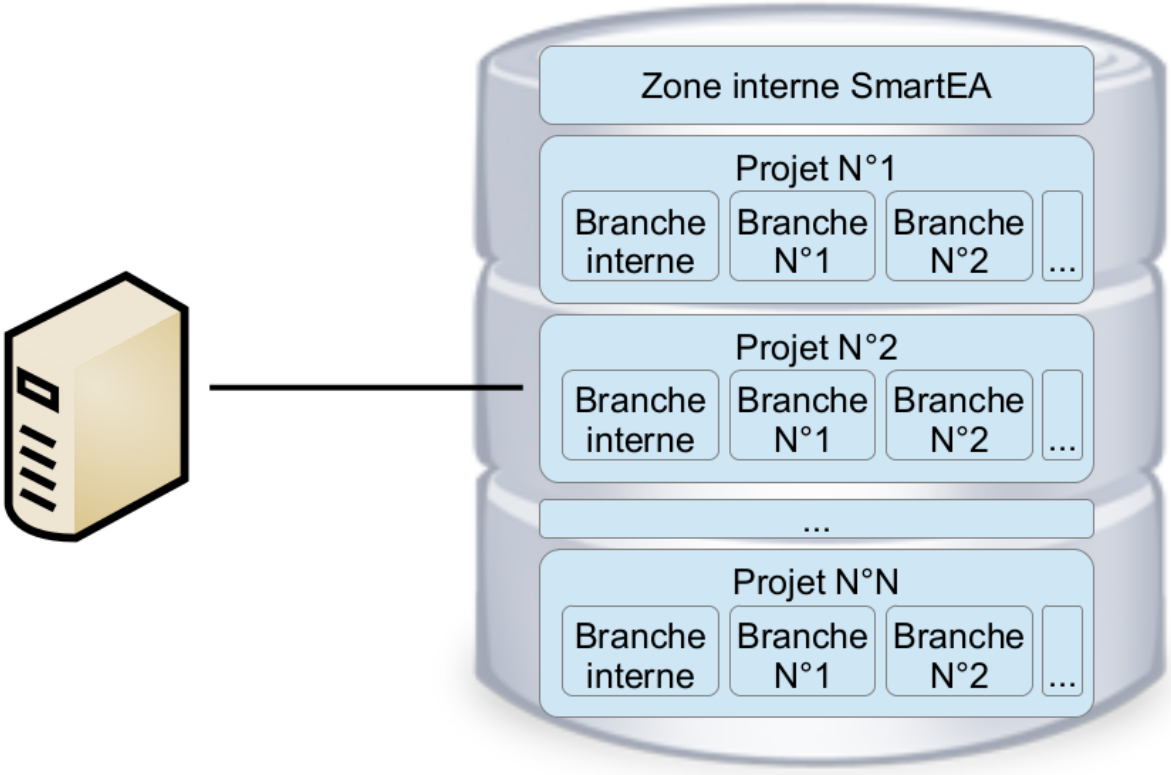
Côté serveur, l'application est hébergée sur un serveur d'application Jetty (déployé dans un conteneur OSGi).

Les données sont persistées dans une base de données relationnelle. SmartEA est homologué sur la base de données PostgreSQL (cf. pré-requis techniques pour plus d'informations).



SmartEA a besoin d'un ensemble de schémas de base de données pour fonctionner :

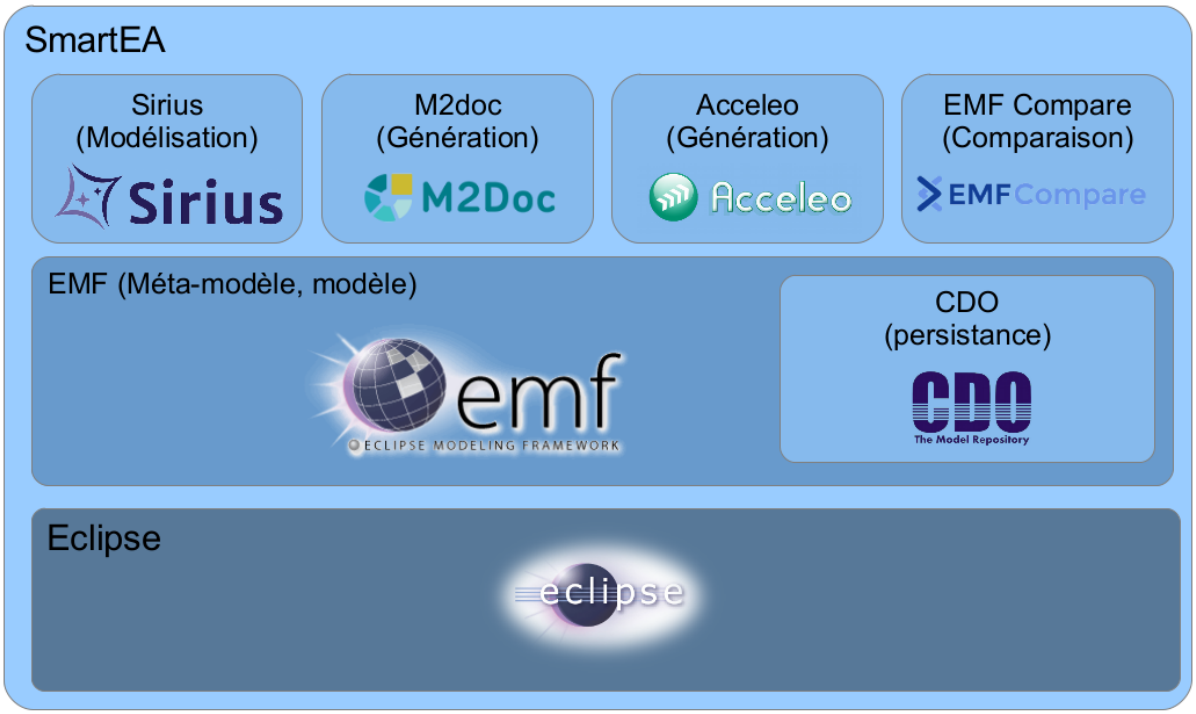
- Un premier schéma pour les données internes à SmartEA
- Puis un schéma par projet



3.2. Architecture du client

Côté client, il est possible d'utiliser soit le modeleur, soit le navigateur web.

Le modeleur est une application RCP qui nécessite la présence d'une machine virtuelle Java installée (cf. pré-requis techniques pour plus d'informations). Ce client communique avec le serveur en utilisant le port HTTP/HTTPS.



Le navigateur web ne communique avec le serveur qu'au travers du port HTTP/HTTPS.

4. Installation de base

Cette section décrit étape par étape l'installation de base du serveur et du modeleur.

Nous allons voir comment :

- démarrer le serveur,
- installer un module,
- configurer l'accès à PostgreSQL,
- configurer l'authentification LDAP, Active Directory ou SSO,
- installer une licence définitive.

4.1. Premier démarrage

Afin de vérifier la configuration logicielle et le réseau, nous allons procéder à un premier démarrage du serveur et du modeleur.

1. Démarrage du serveur

Vous devez disposer d'une archive zip nommée `Obeo-SmartEA-Server-White-Label-<x.y.z>.<qualifier>-<os>.x86_64.zip`.

- `<x.y.z>` est le numéro de version du serveur.
- `<qualifier>` est l'estampille temporelle indiquant le moment de la construction du serveur.
- `<os>` est le système d'exploitation pour lequel le serveur est destiné.

Cette archive est à décompresser sur une machine identifiée pour héberger le serveur. Cette machine doit respecter les pré-requis techniques.

Important : Sous Windows, vous devez décompresser l'archive dans un répertoire dont le chemin ne contient aucun caractère `espace`.

Important : Il est recommandé de permettre au serveur d'utiliser au minimum 2Go de RAM (à adapter selon la taille de votre référentiel). Pour cela, ouvrez le fichier `Obeo.smartEA-Server.ini` avec un éditeur de texte et remplacez `-Xmx512m` par `-Xmx2048m`.

Important : Vous ne devez pas renommer le fichier `Obeo.smartEA-Server.exe`. Si vous renommez ce fichier, les paramètres se trouvant dans le fichier `Obeo.smartEA-Server.ini` ne seront plus pris en compte et le serveur ne fonctionnera pas correctement.

Une fois l'archive décompressée et le paramètre `-Xmx` se trouvant dans le fichier `Obeo.smartEA-Server.ini` modifié, nous allons configurer un projet minimal permettant de démarrer le serveur. Pour cela :

- Ouvrez avec un éditeur de texte le fichier `etc/projects/voyagediscount.conf`
- Désactivez le projet en remplaçant `enabled=true` par `enabled=false`
- Sauvegardez
- Ouvrez avec un éditeur de texte le fichier `etc/projects/sanbox.conf`
- Commentez les lignes
  - `default.path.prism=defaults/voyagediscount/MAIN/data.prism`
  - `default.path.semantic=defaults/archimate.semantic`
- Décommentez les lignes
  - `default.path.prism=defaults/default.prism`
  - `default.path.semantic=defaults/default.semantic`
- Sauvegardez

Vous pouvez désormais démarrer le serveur en ligne de commande avec la commande suivante :

```
* obeo-smartea-server.bat start # Windows
```

```
* obeo-smartea-server.sh start # Unix
```

2. Vérification du serveur

Nous allons maintenant vérifier que le serveur est bien démarré en utilisant un navigateur web.

Si vous avez accès à un navigateur web depuis le serveur, vous pouvez utiliser l'URL `localhost:8080`.Sinon, utilisez l'IP (par exemple 192.168.30.1) ou le nom du serveur et testez que `192.168.30.1:8080` est bien disponible.

Si le serveur est démarré, vous accédez à la page de connexion.

En vous identifiant avec `admin` et le mot de passe `admin`, vous accédez à la page d'accueil.

Vous pouvez consulter les traces du serveur dans `/data/logs/smartea.log`.

Si une erreur ne permet pas le démarrage du serveur, vous trouverez la trace dans `/configuration/xxxxxxxxxxx.log`.

Que faire si le serveur ne démarre pas ?

3. Démarrage du modeleur

Vous devez disposer d'une archive zip nommée `Obeo-SmartEA-Modeler-<x.y.z>.<qualifier>-<os>.x86_64.zip`.

- `<x.y.z>` est le numéro de version du modeleur.
- `<qualifier>` est l'estampille temporelle indiquant le moment de la construction du modeleur.
- `<os>` est le système d'exploitation pour lequel le modeleur est destiné.

Cette archive est à décompresser sur une machine respectant les pré-requis techniques.

Si le serveur est lancé sur votre poste vous pouvez directement utiliser le modeleur (aucune configuration nécessaire).Sinon, vous devez éditer le fichier `application.properties`. Remplacez la valeur `localhost` de la propriété `webServer` par l'IP ou le nom du serveur.

Important : Il est recommandé de permettre au modeleur d'utiliser au minimum 2Go de RAM. Pour cela, ouvrez le fichier `Obeo.smartEA-Modeler.ini` avec un éditeur de texte et remplacez `-Xmx1024m` par `-Xmx2048m`.

Important : Vous ne devez pas renommer le fichier `Obeo.smartEA-Modeler.exe`. Si vous renommez ce fichier, les paramètres se trouvant dans le fichier `Obeo.smartEA-Modeler.ini` ne seront plus pris en compte et le modeleur ne fonctionnera pas correctement.

Important : Sous Windows, vous devez décompresser l'archive dans un répertoire dont le chemin ne contient aucun caractère `accentué`.

Pour démarrer le modeleur, double cliquez sur `Obeo.smartEA-Modeler.exe`. Le modeleur se lance et affiche la page de connexion. Entrez l'identifiant `admin` et le mot de passe `admin` afin de vous connecter et accéder à la page d'accueil.

Que faire si le modeleur ne démarre pas ?

Note : Il est possible d'utiliser le modeleur avec une solution de bureau à distance

4. Arrêt du modeleur et du serveur

Une fois les vérifications faites et si celles-ci sont concluantes, stoppez le modeleur en le fermant tout simplement comme tout autre application.

Stoppez ensuite le serveur. Pour cela, utilisez le script utilisé pour le démarrer en passant en argument la commande `stop` :

```
obeo-smartea-server.bat stop # Windows
obeo-smartea-server.sh stop # Unix
```

4.2. Création d'un projet

Maintenant que nous avons vérifié que le serveur et le modeleurs peuvent démarrer et communiquer, nous allons créer un projet qui va accueillir des données.

SmartEA permet de créer un ou plusieurs projets. Par défaut, un projet `sandbox` est déclaré dans le répertoire `etc/projects` du serveur.

L'extension du fichier permettant de configurer le projet est `.conf` et le nom du fichier – sans l'extension – sera utilisé comme identifiant de projet.

Important : Cet identifiant doit être en caractères minuscules et sans caractères spéciaux.

Nous allons maintenant créer un nouveau projet.

Copiez le fichier `sandbox.conf`. Pour l'exemple, nous utiliserons `monprojet` comme identifiant et donc comme nom de fichier.

Copiez également l'image `sandbox.png` du projet (`monprojet.png`). Vous pourrez la remplacer ultérieurement.

Editez ensuite le fichier `etc/projets/monprojet.conf` de la manière suivante :

```
# Information générale sur le projet
label=Mon Projet
description=Mon premier projet SmartEA
enabled=true

# Configuration de la base de données
db.url=jdbc:h2:file:${smartea.data.directory}/run/content/monprojet/h2/
monprojetdb;DB_CLOSE_ON_EXIT=FALSE
db.driver=org.h2.Driver
db.user=sa
db.pass=
db.pool.timeout=15000
db.pool.maxSize=30
db.pool.minSize=1
db.pool.maxIdleTimeExcessConnections=0
sql.request.maxLogicalIdsRetrievedByRequest=1

# Prisme et modèle sémantique par défaut
default.path.prism=defaults/default.prism
default.path.semantic=defaults/default.semantic
```

Si le projet est déstiné à être utilisé avec le module Archimate, vous pouvez remplacer les deux derniers paramètres par :

```
default.path.prism=defaults/archimate.prism
default.path.semantic=defaults/archimate.semantic
```

Ainsi le prisme par défaut sera déjà paramétré pour afficher les éléments archimate, et le modèle sémantique par défaut comprendra des répertoires (vides) destinés à recevoir les éléments des différentes couches d'archimate.

Pour plus de détails sur les paramètres de ce fichier, consultez Description de l'intégralité des paramètres

Le nouveau projet est maintenant configuré.

4.3. Installation d'un module SmartEA

Nous allons voir dans cette section comment installer un module.

SmartEA est composé d'un socle générique sur lequel sont installés des modules. Un module fournit :

- un ou des métamodèles,

- des représentations Sirius,
- des générateurs Acceleo/Java,
- des extensions pour le serveur et/ou les modeleurs.

Des modules prédéfinis sont livrés avec SmartEA : Archimate, BPMN, RGPD, etc.. Il est également possible de définir son propre module, vous trouverez plus d'informations à ce sujet dans le Guide de personnalisation.

Un module est un update-site Eclipse. C'est un répertoire contenant :
- un répertoire `features`
- un répertoire `plugins`
- un ficher `artifacts.jar`
- un ficher `content.jar`

Nous allons maintenant installer le module Archimate.

Vous devez disposer d'une archive zip nommée `Obeo-SmartEA-Module-Archimate-<x.y.z>.<qualifier>.zip` :
- `<x.y.z>` est le numéro de version du module.
- `<qualifier>` est l'estampille temporelle indiquant le moment de la construction du module.

Cette archive est à décompresser sur la machine où est déployé le serveur.

Les éléments décompressés (fichiers et répertoires) sont ensuite à copier dans le répertoire `modules` du serveur dans **un dossier dédié**. Pour l'exemple nous le nommerons `archimate`. Nous avons donc l'arborescence suivante : `modules/archimate/`.

Une fois l'update site copié, le module est dit déployé. Au prochain démarrage du serveur, le module déployé dans le répertoire `modules` s'installera automatiquement sur le serveur. De même, au prochain démarrage du Modeleur, une procédure d'installation automatisée se déclenchera. Le module sera installé sur le Modeleur.

La prochaine étape consiste à remettre à zéro de la base de données du projet `monprojet` car nous allons initialiser la base de données automatiquement avec des données d'exemple. Etant donné que nous utilisons une base de données H2 sur disque vous devez simplement supprimer le dossier `/data/run/content/monprojet`.

SmartEA utilise la notion de prisme pour gérer les droits d'accès au référentiel. Il est possible d'associer un ou plusieurs prismes a un utilisateur donné (voir Guide d4administration pour plus d'information).

Afin de faciliter le démarrage, nous allons initialiser un prisme déjà configuré pour le module `archimate` ainsi qu'un modèle d'exemple qui permettra d'initialiser le référentiel. Pour ce faire, éditez le fichier `etc/projets/monprojet.conf` :

```
default.path.prism=defaults/archimate.prism
default.path.semantic=defaults/archimate.semantic
```

Ces deux fichiers sont présents dans le répertoire `/data/defaults/`

Le module est maintenant installé.

Vous pouvez démarrer le serveur, puis vous connecter sur le référentiel à l'aide d'un navigateur. Connectez vous ensuite sur le référentiel en utilisant le Modeleur. Vous pouvez désormais créer des éléments Archimate et des diagrammes Archimate.

En cas de mauvaise configuration,
- stoppez le serveur
- effacez ${smartea.data.directory}/run/content/monprojet/h2/monprojet
- modifiez votre configuration
- puis redémarrez le serveur.

4.4. Configuration de l'accès aux bases de données

Nous allons maintenant voir comment configurer les informations d'accès aux bases de données utilisées par le serveur SmartEA.

Pour simplifier le démarrage, le serveur utilise une base de données H2. Cette base de données est suffisante et recommandé lors de la phase de développement d'un module.

Important : Lorsque le serveur est déployé dans un environnement de **pré-production** ou **production** nous recommandons d'utiliser une base de données PostgreSQL gérée par votre SI. Il est nécessaire qu'une sauvegarde régulière de la base de données soit mise en place. Vous pouvez mettre en place une solution de sauvegarde et de restauration de base de données s'intégrant à vos processus. La configuration de la base de données est décrite dans la Section Configurer le serveur avec PostgreSQL.

La configuration des informations JDBC permettant d'accéder à la base de données sont dans 2 types de fichiers de configuration :

1. Le fichier `etc/application_internaldb.conf` contient les paramètres JDBC permettant d'accéder à la base interne. Cette base est nécessaire au démarrage de SmartEA.
2. Les fichiers `etc/projets/<id_projet>.conf` contient les paramètres JDBC permettant d'accéder à la base du projet `<id_projet>`.

Description de l'intégralité des paramètres

La remise à zéro de la base de données (à utiliser avec précaution !) doit être faite serveur SmartEA éteint. Pour cela, suivez la procédure de votre base de données.

Pour une base de données H2, la remise à zéro peut être faite par suppression du répertoire qui se trouve dans l'URL JDBC.

4.5. Configuration de l'authentification

Pour simplifier le démarrage, le serveur gère l'authentification des utilisateurs par l'intermédiaire d'un fichier texte : `etc/users.yml`.

Important : Lorsque le serveur est déployé dans un environnement de **pré-production** ou **production** nous recommandons une authentification par l'intermédiaire d'un annuaire LDAP ou Active Directory ou d'utiliser un site tiers mettant en oeuvre l'authentification SSO.

La configuration du Serveur SmartEA pour déléguer l'authentification à un annuaire LDAP ou Active Directory est décrite dans la Section Authentification LDAP/Active Directory.

La configuration du Serveur SmartEA pour déléguer l'authentification à un site tiers SSO est décrite dans la Section Authentification SSO.

Ci-dessous vous trouverez un exemple de fichier `etc/users.yml`:

```
admin:
    admin: true
    password: admin
    firstName: Admin
    lastName: ADMIN
    prismIDs: [EntArch]
```

Les différents paramètres sont décrits dans la Section présentant le fichier user.yml

Pour ajouter de nouveaux utilisateurs, vous pouvez copier coller l'exemple.**Attention à bien respecter la syntaxe YML. Les espaces et le retour à la ligne ont une importance.**

4.6. Installation d'une licence

Le serveur contient une licence d'évaluation d'une validité de deux mois. Elle doit être remplacée.

Pour obtenir une licence, vous devez suivre le protocole ci-dessous.

1. Obtention d'une licence

Les actions suivantes doivent être exécutées sur la machine qui hébergera le serveur SmartEA, par l'utilisateur qui démarrera le serveur SmartEA.

En ligne de commande, placez-vous dans le répertoire d'installation du serveur. Il contient les scripts de génération de clé :

- `obeo-smartea-server-licensekey.bat` pour Windows
- `obeo-smartea-server-licensekey.sh` pour Unix

Il vous suffit de lancer le script correspondant à votre système d'exploitation, puis de récupérer le résultat obtenu dans la console :

```
smartea@prodserver:/opt/smartea/Obeo-SmartEA-Server$ ./obeo-smartea-server-licensekey.sh
Welcome to Obeo SmartEA Server Registration.

Could you please send a mail to registration@obeo.fr with the following informations :
* Organization
* First Name
* Last Name
* Phone Number
* The following registration ID : 0a79879d0d6a7a472fc163a8d349afd9fe[...]8bda3a9335
smartea@prodserver:/opt/smartea/Obeo-SmartEA-Server$
```

Une fois le `Registration ID` généré :

- Copiez le résultat affiché dans la console à partir de `* Organization` jusqu'à la fin du `Registration ID`,
- Collez le résultat dans un email adressé à registration@obeo.fr. **Attention :** certains logiciels de messagerie tronquent l'information copiée/collée. Si c'est le cas, copiez/collez les informations dans un fichier texte que vous joindrez à votre email.
- Transmettez l'email après avoir renseigné les champs Organisation, Nom, Prénom et Téléphone.

Obeo vous transmettra en retour un fichier nommé `smartEALicense.key.zip`.

Important : Si votre distribution Linux ne fournit pas la commande `ifconfig` (distribution récente de Linux), le script `obeo-smartea-server-licensekey.sh` ne peut générer correctement le `Registration ID`.Dans ce cas vous devez fournir à registration@obeo.fr l'adresse MAC du serveur Obeo SmartEA ainsi que le login de l'utilisateur qui execute le serveur SmartEA.

2. Installation de la licence

Pour installer une nouvelle licence :

- Arrêtez le serveur si celui-ci est démarré.
- Décompressez le fichier `smartEALicense.key.zip` envoyé par Obeo.
- Remplacez le fichier `smartEALicense.key` obtenu dans le répertoire `etc/` du serveur.
- Redémarrez le serveur.

Pour vérifier que la licence est bien installée, vous pouvez consulter la page d'administration

4.7. Configurations supplémentaires

Lorsque le serveur est déployé dans un environnement de **pré-production** ou **production** nous préconisons :

- d'utiliser PostgreSQL comme support de persistance.
- d'utiliser un annuaire LDAP ou Active Directory pour l'authentification.
- de gérer le lancement du serveur sous forme de service, que ce soit sous Windows ou Unix.
- d'activer le protocole HTTPS pour sécuriser les échanges

5. Installation avancée

5.1. Configuration de PostgreSQL

Important : Assurez vous que votre installation de PostgreSQL respecte les pré-requis.

Cette Section présente la création des schémas et des utilisateurs PostgreSQL ainsi que les configurations du serveur SmartEA associées.

Le serveur SmartEA a besoin d'au moins 2 schémas PostgreSQL pour fonctionner :

1. Un schéma pour son fonctionnement interne. Ce schéma est nommé `smarteainternal` (utilisez uniquement des minuscules)
2. Un schéma pour chaque projet. Dans la suite nous n'avons qu'un projet (`monprojet`) et donc qu'un seul schéma nommé `monprojet`.

Les schémas peuvent appartenir à la même base de données PostgreSQL (recommandé) ou a des bases de données PostgreSQL différentes.

Dans la suite, nous présentons le cas où les schémas appartiennent à la même base de données PostgreSQL. La méthode conseillée est donc de :

- Créer une base de données (ou utiliser une base existante)
  - Créer dans cette base un schéma nommé `smarteainternal` pour le fonctionnement interne de SmartEA avec un utilisateur spécifique
  - Créer dans cette base un schéma nommé `monprojet`, avec un utilisateur spécifique

Pour créer la base, les utilisateurs et les schémas, vous pouvez utiliser l'interface graphique de PostgreSQL `PgAdmin` ou lancer les commandes en utilisant le shell de PostgreSQL.La suite de cette Section présente les instructions à utiliser dans le shell.

Nous utiliserons les variables suivantes dans les commandes :

- `smarteadb` : Le nom de la base de données
- `smarteainternal` : Le nom du schéma et de l'utilisateur pour le fonctionnement interne de SmartEA
- `monprojet` : Le nom du schéma et de l'utilisateur pour le projet
- `pwd` : le mot de passe SQL. La procédure qui suit utilise le même pour tous les schémas/utilisateurs mais il est possible de les différencier. Ce mot de passe sera nécessaire dans les fichiers de configuration du serveur SmartEA.

1. Configuration du serveur PostgreSQL

Avant de démarrer le serveur PostgreSQL, configurez le afin qu'il prenne en compte les spécificités de votre machine : nombre de processeurs, mémoire, etc. Pour cela, vous pouvez vous aider du configurateur disponible ici : https://pgtune.leopard.in.ua/#/.

Attention également au paramètre `idle_in_transaction_session_timeout`.Mettez une valeur suffisament grande pour que les sessions ne soient pas fermées automatiquement après une nuit ou un week-end.Il est possible de mettre la valeur `0` afin d'indiquer au serveur PostgreSQL de ne pas fermer les sessions inactives.

Afin d'optimiser les performances, vous pouvez également positionner `synchronous_commit` à la valeur `off`. Ce paramètre indique si la validation des transactions doit attendre l'écriture des enregistrements WAL avant que la commande ne renvoie une indication de réussite au client.Quand ce paramètre est désactivé (`off`), il peut exister un délai entre le moment où le succès est rapporté et le moment où la transaction est vraiment protégée d'un arrêt brutal du serveur (le délai maximum est de trois fois `wal_writer_delay`).La configuration de ce paramètre à `off` n'implique aucun risque d'incohérence dans la base de données : un arrêt brutal du système d'exploitation ou d'une base de données peut résulter en quelques transactions récentes prétendument validées perdues malgré tout.Cependant, l'état de la base de données est identique à celui obtenu si les transactions avaient été correctement annulées.C'est pourquoi la désactivation de `synchronous_commit` est une alternative utile quand la performance est importante. Documentation PostgreSQL : https://www.postgresql.org/docs/13/wal-async-commit.html

scram-sha-256 :

Si PostgreSQL ne se trouve pas sur la même machine que le serveur SmartEA il est fortement recommandé d'activer la méthode d'authentification scram-sha-256 : https://www.postgresql.org/docs/current/auth-password.html#:~:text=The%20method%20scram%2Dsha%2D256,is%20thought%20to%20be%20secure

```
The method scram-sha-256 performs SCRAM-SHA-256 authentication, as described in RFC 7677. It
 is a challenge-response scheme that prevents password sniffing on untrusted connections and
 supports storing passwords on the server in a cryptographically hashed form that is thought
 to be secure.
This is the most secure of the currently provided methods, but it is not supported by older
 client libraries.
```

SmartEA supporte la méthode d'authentification scram-sha-256 :

- 1. Activer le scram-sha-256 au niveau du SGBD :
  - Ouvrez en édition le fichier `postgresql.conf`.
  - Modifiez de la manière suivante le paramètre `password_encryption = scram-sha-256` (par défaut la valeur est `md5`).
  - Sauvegardez.
  - Redémarrez le serveur PostgreSQL.
- 2. Activer le scram-sha-256 pour les connexions entrantes
  - Ouvrez en édition le fichier `pg_hba.conf`.
  - Remplacez `md5` et `trust` par `scram-sha-256` dans la colonne `METHOD`.
  - Sauvegardez.
- 3. Mettre à jour les mots de passe des utilisateurs Postgresql existants
  - Vérifiez si les mots de passe existants sont encodés avec la méthode `md5` : `select rolname,rolpassword from pg_authid;`.
  - Si dans la colonne `rolpassword` vous ne voyez pas `SCRAM-SHA-256*` vous devez regénérer les mots de passe : `ALTER USER voyagediscount with encrypted password '123';`.
  - Revérifiez que les mots de passe sont correctement encodés : `select rolname,rolpassword from pg_authid;`. Dans la colonne `rolpassword` les lignes doivent commencer par `SCRAM-SHA-256*`.
- 4. Lancement du serveur SmartEA
  - Démarrez le serveur SmartEA. Aucune modification n'est nécessaire.

2. Démarrer le serveur PostgreSQL

Sous Linux, il suffit de lancer le service :

```
sudo /etc/init.d/postgresql start
```

Sous Windows, il suffit de lancer le SQL shell fourni avec PostgreSQL.

3. Ouvrir la console PostgreSQL

Sous Linux, vous pouvez utiliser la commande suivante :

```
sudo -u postgres /Library/PostgreSQL/10.4/bin/psql
```

Sous Windows, utiliser le shell mentionné ci-dessus.

4. Créer la base de données

Vous pouvez utiliser une base de données existante ou en créer une nouvelle (nommée «smarteadb» par exemple).

L'utilisateur courant doit pouvoir créer des bases et des rôles (par exemple le superutilisateur root, par défaut postgres).

Dans la console SQL, entrez la commande suivante :

```
CREATE DATABASE smarteadb;
```

Par la suite déconnectez vous, puis reconnectez vous sur la base de données (relancez le shell sous Windows, en spécifiant le nom de la base de données).

```
sudo /Library/PostgreSQL/10.4/bin/psql smarteadb
```

5. Créer les schémas et les utilisateurs

Nous allons créer un utilisateur et un schéma associé pour `smarteainternal` et `monprojet` dans la console SQL. Pour simplifier, il est conseillé d'utiliser le même nom pour un schéma et son utilisateur associé, soit {schema_user_name} = {schema_name}.

N'oubliez pas d'utiliser ici un mot de passe sécurisé.

```
CREATE USER smarteainternal;
ALTER USER smarteainternal with encrypted password 'pwd';
CREATE SCHEMA smarteainternal AUTHORIZATION smarteainternal;
ALTER ROLE smarteainternal IN DATABASE smarteadb SET search_path = smarteainternal;
```

```
CREATE USER monprojet;
ALTER USER monprojet with encrypted password 'pwd';
CREATE SCHEMA monprojet AUTHORIZATION monprojet;
ALTER ROLE monprojet IN DATABASE smarteadb SET search_path = monprojet;
```

Important : Si vous avez utilisé Amazon RDS (Amazon Relational Database Service) pour créer une instance PostgreSQL, vous devez ajouter les lignes suivantes après les lignes `ALTER USER ...`.

```
GRANT smarteainternal TO aws_postgres;
```

et

```
GRANT monprojet TO aws_postgres;
```

6. Mettre à jour la configuration du serveur SmartEA

Nous allons à présent mettre à jour la configuration du serveur.

Vous devez connaître l'adresse de la base de données, qui peut être sur la même machine que le serveur SmartEA.

Pour la partie interne de SmartEA, vous devez éditer le fichier `etc/application_internaldb.conf` comme ci-dessous :

```
db.url=jdbc:postgresql://localhost:5432/smarteadb
db.driver=org.postgresql.Driver
db.user=smarteainternal
db.pass=pwd
```

Pour le projet, vous devez éditer le fichier `etc/projects/monprojet.conf` comme ci-dessous :

```
db.url=jdbc:postgresql://localhost:5432/smarteadb
db.driver=org.postgresql.Driver
db.user=monprojet
db.pass=pwd
```

Vous pouvez à présent relancer le serveur SmartEA et vérifier qu'il n'y a pas d'erreur.

7. Optimiser les tables des relations

Lorsque les tables de la base de données portant les objets de relation prennent une certaine ampleur (de l'ordre de ~100 000 entrées), il se peut que des problèmes de performance soient observés lorsque ces relations sont recherchées, notamment sur le widget Marguerite des pages de détails des objets.

Pour y remédier il faut créer des index sur ces tables pour permettre une recherche rapide des relations. Ces index doivent porter sur la colonne CDO_CREATED ainsi que sur la colonne portant la cible de la relation (par exemple TARGET pour les relations ArchiMate unidirectionnelles).

Pour le métamodèle ArchiMate, pour chaque schéma concerné, voici les commandes de création des index :

```
CREATE INDEX artifact_artifact_inverse_index ON <schemaname>.artifact_artifact(cdo_created,
 context);
CREATE INDEX archimate_accessrelationship_inverse_index ON
 <schemaname>.archimate_accessrelationship(cdo_created, target);
CREATE INDEX archimate_triggeringrelationship_inverse_index ON
 <schemaname>.archimate_triggeringrelationship(cdo_created, target);
CREATE INDEX archimate_realizationrelationship_inverse_index ON
 <schemaname>.archimate_realizationrelationship(cdo_created, target);
CREATE INDEX archimate_servingrelationship_inverse_index ON
 <schemaname>.archimate_servingrelationship(cdo_created, target);
CREATE INDEX archimate_aggregationrelationship_inverse_index ON
 <schemaname>.archimate_aggregationrelationship(cdo_created, target);
CREATE INDEX archimate_assignmentrelationship_inverse_index ON
 <schemaname>.archimate_assignmentrelationship(cdo_created, target);
CREATE INDEX archimate_flowrelationship_inverse_index ON
 <schemaname>.archimate_flowrelationship(cdo_created, target);
CREATE INDEX archimate_associationrelationship_inverse_index ON
 <schemaname>.archimate_associationrelationship(cdo_created, end2);
CREATE INDEX archimate_compositionrelationship_inverse_index ON
 <schemaname>.archimate_compositionrelationship(cdo_created, target);
CREATE INDEX archimate_influencerelationship_inverse_index ON
 <schemaname>.archimate_influencerelationship(cdo_created, target);
```

Si d'autres types de relations sont largement utilisés dans votre référentiel il est conseillé de créer des index sur leurs tables d'après ce modèle.

Retrouver les problèmes connus dans la FAQ

Autres commandes utiles

1. Arrêter le serveur PostgreSQL

```
sudo /etc/init.d/postgresql stop
```

2. Fermer la console PostgreSQL

```
\q
```

3. Vider la base de données PosgreSQL

```
drop schema {schema_name} cascade;
create schema {schema_name};
exit;
```

Attention, l'exécution de cette commande supprimera toutes les données de la base SmartEA.

4. Perte du mot de passe de l'utilisateur `postgres`

Si vous n'avez plus le mot de passe de l'utilisateur `postgres`, vous pouvez le réinitialiser en appliquant la procédure décrite ici : https://www.postgresqltutorial.com/postgresql-reset-password/

5.2. Installation d'un service démarrant et stoppant le serveur SmartEA

Que ce soit sous Windows ou Unix, nous vous conseillons de gérer le lancement du serveur SmartEA sous forme de service.

5.2.1. Service Windows

Le serveur SmartEA Windows intègre un utilitaire permettant son installation en tant que service. La procédure permettant de l'exploiter est la suivante :

1. ouvrez une invite de commande Windows en tant qu'administrateur
2. positionnez vous dans le répertoire `service` du serveur SmartEA
3. installez le service à l'aide de la commande `ObeoSmartEAService.exe install`

Une fois le service installé, il apparaît dans l'outil Windows d'administration de services sous le nom **Obeo SmartEA Service**.

Vous pouvez le démarrer ou l'arrêter à partir de là, ou bien utiliser les commandes `ObeoSmartEAService.exe start` / `ObeoSmartEAService.exe stop` dans la même invite de commande que précédemment.

Vous pouvez également désinstaller le service avec la commande `ObeoSmartEAService.exe uninstall`.

Remarques :

- il est recommandé de laisser le serveur démarrer/s'arrêter jusqu'au bout (page d'accueil disponible/non trouvée) avant de lancer une nouvelle commande avec l'utilitaire.
- les logs relatifs au lancement en tant que service sont disponibles dans le répertoire `data/logs/service` du serveur.
- si le serveur ne démarre pas, il peut y avoir un problème avec la version de java par défaut de windows. Celle-ci doit être en 64 bits.
- si le service Windows est dans un état indéterminé (empêchant son démarrage, arrêt ou désinstallation), vous pouvez forcer sa désinstallation en suivant la procédure suivante :
  - fermez l'administrateur de services Windows si il est ouvert
  - avec l'invite de commande administrateur : `sc queryex ObeoSmartEAService`
  - dans le résultat récupérez l'identifiant du processus (`PID`)
  - fermez le processus correspondant avec la commande : `taskkill /pid <PID> /f`
  - désinstallez le service : `sc delete ObeoSmartEAService`

Ces commandes agissent seulement sur le service et n'endommageront pas le serveur SmartEA.

5.2.2. Service Unix

Il est possible de démarrer et de stopper le serveur SmartEA par l'intermédiaire d'un service.

5.2.2.1. `init.d`

Ci dessous se trouve la procédure pour Debian 8 et le système `init.d`. Elle est à adapter en fonction de votre distribution de Linux.

Copiez le script suivant dans un fichier nommé `smartea`.

```
#! /bin/bash
### BEGIN INIT INFO
# Provides:          smartea
# Required-Start:    $syslog postgresql
# Required-Stop:     $syslog postresql
# Should-Start:      $network $named $time
# Should-Stop:       $network $named $time
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Start and stop the Smartea Server
# Description:       Service to start and stop the SmartEA Server.
### END INIT INFO
#
# You may want to temporarely remove the >/dev/null for debugging purposes
#
# Path to SmartEA Server install
SMARTEA_HOME=/Shares/INTERNET/smartea/Obeo-SmartEA-Server
SMARTEA_CMD=$SMARTEA_HOME/obeo-smartea-server.sh
# Path to the JVM
```

```
PATH=$PATH:/opt/java/jdk/jre/bin/
# User running the SmartEA Server process
USER=integration
cd $SMARTEA_HOME
sudo -u $USER export $PATH ; $SMARTEA_CMD $1
exit 0
```

Si nécessaire, modifiez le script afin de l'adapter à votre contexte.

Note : si vous avez copié/collé le fichier depuis un environnement Windows vers un environnement Linux, il se peut que votre fichier contienne des retours chariot au format DOS. Vous pouvez le vérifier avec la commande :

```
 cat -v smartea
```

Si les caractères en fin de ligne sont `^M` cela signifie que les retours chariot sont au format DOS. Vous devez les modifier en utilisant la commande :

```
sudo sed -i -e 's/\r//g' smartea
```

Copiez ensuite le script dans le répertoire /etc/init.d et définissez les bons droits.

```
sudo cp ~/smartea /etc/init.d/.
sudo chmod 0755 /etc/init.d/smartea
```

Rechargez les daemons du système.

```
sudo systemctl daemon-reload
```

Démarrez le service.

```
sudo /etc/init.d/smartea start
```

Vérifiez que le serveur SmartEA est bien démarré.

Stoppez le service.

```
sudo /etc/init.d/smartea stop
```

Ajoutez le service au démarrage.

```
sudo update-rc.d smartea defaults
```

Pour finir de tester, stoppez puis redémarrez la machine afin de vérifier que le service démarre et stoppe correctement.

5.2.2.2. `systemd` :

Si votre distribution Linux utilise `systemd`, voici un exemple de procédure à adapter à votre environnement.

Copiez le fichier de configuration suivant dans un fichier nommé `smartea.service` (répertoire `/etc/systemd/system/`).

```
[Unit]
Description=Smartea server
After=syslog.target postgresql.target
[Service]
Type=simple
User=smartea
#EnvironmentFile=/etc/systemd/system/smartea.env
WorkingDirectory=/Shares/INTERNET/smartea/Obeo-SmartEA-Server/
ExecStart=/Shares/INTERNET/smartea/Obeo-SmartEA-Server/obeo-smartea-server.sh start
ExecStop=/Shares/INTERNET/smartea/Obeo-SmartEA-Server/obeo-smartea-server.sh stop
[Install]
WantedBy=multi-user.target
```

Selon votre distribution Linux et sa configuration, vous pouvez être amené à remplacer `After=syslog.target postgresql.target` par `Requires=rsyslog.service postgresql.target`.

Vous devez ensuite modifier le fichier `obeo-smartea-server.sh` de la manière suivante :

```
#!/bin/bash
DIRECTORY=/Shares/INTERNET/smartea/Obeo-SmartEA-Server
export PATH=/Shares/TOOLS/java/jdk/jre/bin/:$PATH
....
```

Rechargez les daemons du système.

```
sudo systemctl daemon-reload
```

Démarrez le service.

```
systemctl start smartea.service
```

A chaque modification, vous devez recharger les daemons du système et relancer le service smartea :

```
systemctl daemon-reload
systemctl restart smartea.service
```

Pour stopper le service, utilisez la commande stop :

```
systemctl stop smartea.service
```

Activez le service, de façon à ce qu'il soit lancé au démarrage :

```
systemctl enable smartea.service
```

Pour finir de tester, stoppez puis redémarrez la machine afin de vérifier que le service démarre et stoppe correctement.

Pour désactiver le service, utilisez la commande disable :

```
systemctl disable smartea.service
```

5.3. Utilisation du protocole HTTPS

Pour sécuriser les échanges client/serveur il est possible d'utiliser le protocole HTTPS. Il est alors nécessaire de configurer le serveur et le modeleur.

Coté serveur, le HTTPS peut être configuré de deux manières :

- Utilisation du serveur HTTPS embarqué de Jetty,
- Installation d'un serveur web «frontal» devant le serveur Jetty. Il est par exemple possible d'utiliser un serveur frontal Apache.

5.3.1. Certificat

Important : ce paragraphe décrit une procédure permettant de générer un certificat et de l'installer à des fins de test. Il n'a pas vocation à servir de modèle ni à se substituer aux pratiques de gestion des certificats en vigueur dans votre société.

Dans notre exemple nous allons générer un keystore nommé `certificate.jks` en utilisant un mot de passe basique pour le modifier (`secret`). Le certificat lui-même sera créé avec le même mot de passe. Nous utiliserons l'alias `smartea` pour le référencer dans le keystore.

Si votre modeleur accédera à votre serveur par un nom de machine (ex : `myhost` dans notre exemple), lancez la commande `keytool` (la commande keytool se trouve dans le répertoire bin de votre jre: `jre/bin/keytool`) et spécifiez le nom de machine à la première question posée :

```
conf $ keytool -genkey -alias smartea -keyalg RSA -keypass secret -storepass secret -keystore
 certificate.jks
What is your first and last name?
  [Unknown]:  myhost
What is the name of your organizational unit?
  [Unknown]:
What is the name of your organization?
  [Unknown]:
What is the name of your City or Locality?
  [Unknown]:
What is the name of your State or Province?
  [Unknown]:
What is the two-letter country code for this unit?
  [Unknown]:
Is CN=myhost, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=Unknown correct?
  [no]:  yes
```

Si votre modeleur accédera à votre serveur par une adresse IP, ajoutez simplement le suffixe `-ext san=ip:<my-ip-adress>` à la commande `keytool` (remplacez `192.16.1.10` par votre valeur dans l'exemple ci-dessous) :

```
conf $ keytool -genkey -alias smartea -keyalg RSA -keypass secret -storepass secret -keystore
 certificate.jks -ext san=ip:192.168.1.10
What is your first and last name?
  [Unknown]:
What is the name of your organizational unit?
  [Unknown]:
What is the name of your organization?
  [Unknown]:
What is the name of your City or Locality?
  [Unknown]:
What is the name of your State or Province?
  [Unknown]:
What is the two-letter country code for this unit?
  [Unknown]:
Is CN=Unknown, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=Unknown correct?
  [no]:  yes
```

Dans un cas comme dans l'autre vous obtenez un fichier `certificate.jks`.

Exporter ensuite le certificat :

```
conf $ keytool -export -alias smartea -file smartea.cer -keystore certificate.jks
Enter keystore password:
Certificate stored in file <smartea.cer>
```

Le fichier `smartea.cer` est maintenant créé.

5.3.2. Configuration du serveur SmartEA

Important : il est supposé ici que vous disposez d'un certificat SSL valide :

- soit le certificat généré au paragraphe précédent (à des fins de tests),
- soit un certificat obtenu selon les pratiques de gestion des certificats en vigueur dans votre société.

Comme dit en introduction, le mode HTTPS peut être activé de deux manières :

- En utilisant le serveur HTTPS embarqué de Jetty,
- En installant un serveur web «frontal» devant le serveur Jetty : Apache par exemple.

5.3.2.1. Utilisation du serveur HTTPS Jetty embarqué

Il est possible d'activer le serveur HTTPS embarqué de Jetty simplement en modifiant le fichier `etc/jetty-http.xml`

Commentez la section se trouvant sous le commentaire `Add a HTTP Connector.`, à partir de `<Call name="addConnector">` jusqu'à `</Call>`.

Le bloc peut être commenter en ajoutant `<!--` en début de zone à commenter et `-->` en fin de zone à commenter.

Puis, décommentez la section se trouvant sous le commentaire `Add a HTTPS Connector.`, à partir de `<New id="httpsConfig" class="org.eclipse.jetty.server.HttpConfiguration">` jusqu'à `</Call>`.

Adaptez la configuration du mode https si nécessaire.

Copiez collez les fichiers `certificate.jks` et `smartea.cer` dans le répertoire `etc/`.

Démarrez le serveur.

Connectez vous ensuite à l'aide de votre navigateur sur `https://[host]:8443`

5.3.2.2. Utilisation d'un serveur web frontal

Dans ce mode, le paramétrage HTTPS du serveur se fait au niveau du serveur web frontal choisi (Apache, ...) et selon les modalités spécifiques de celui-ci. Veuillez vous reporter à la documentation de votre frontal pour étudier les modalités d'installation des certificats SSL, des modules requis et la configuration nécessaire.

Note : Vous devez, notamment installer le module `headers` si vous souhaitez activer `Secure` et `HTTPOnly`.

Sous Debian 8 et Apache 2.4 :

```
sudo a2enmod headers
```

Le serveur SmartEA doit être en mode http et configuré pour n'accepter que les requêtes provenant de `127.0.0.1`.

Pour cela, éditez le fichier `etc/jetty-http.xml` et assurez vous que la section se trouvant sous le commentaire `Add a HTTP Connector.` (à partir de `<Call name="addConnector">` jusqu'à `</Call>`) n'est pas commentée et que la section se trouvant sous le commentaire `Add a HTTPS Connector.` (à partir de `<New id="httpsConfig" class="org.eclipse.jetty.server.HttpConfiguration">` jusqu'à `</Call>`) est commentée.

Au niveau du noeud `<Set name="host"></Set>`, ajoutez que Jetty est bindé sur 127.0.0.1 :

```
<Set name="host">127.0.0.1</Set>
```

Editez ensuite le fichier de configuration d'Apache.

Ajoutez la directive `AllowEncodedSlashes On`.

Si vous voulez activer HttpOnly et Secure, ajoutez les lignes suivantes :

```
Header edit Set-Cookie ^((sessionID|auth)=.*)$ $1;HttpOnly;Secure
RequestHeader set X-Forwarded-Proto "https"
```

Si la connexion CDO a lieu par l'intermédiaire du protocole websocket ou websocket sécurisé alors la ligne suivante est nécessaire :

```
RequestHeader edit Cookie ".*auth=\"([a-z0-9]+)\"[^\"]*sessionID=\"([a-z0-9\.]+)\".*" "auth=
\"$1\"; sessionID=\"$2\""
```

Note : Pour plus d'informations à propos de ces deux paramètres, veuillez vous référer à ces liens : https://www.owasp.org/index.php/SecureFlag et https://www.owasp.org/index.php/HttpOnly

Au final votre fichier de configuration Apache doit ressembler à :

```
<VirtualHost *:8443>
   DocumentRoot "/Library/WebServer/Documents"
   ServerName www.example.com:8443
   ServerAdmin you@example.com
   SSLEngine on
```

```
   SSLCertificateFile "/private/etc/apache2/server.crt"
   SSLCertificateKeyFile "/private/etc/apache2/server.key"
   BrowserMatch "MSIE [2-5]" \
            nokeepalive ssl-unclean-shutdown \
            downgrade-1.0 force-response-1.0
   CustomLog "/private/var/log/apache2/ssl_request_log" \
             "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x \"%r\" %b"
   ProxyPass / http://127.0.0.1:8080/
   ProxyPassReverse / http://127.0.0.1:8080/
   AllowEncodedSlashes On
   Header edit Set-Cookie ^((sessionID|auth)=.*)$ $1;HttpOnly;Secure
   RequestHeader set X-Forwarded-Proto "https"
</VirtualHost>
```

Connectez vous ensuite à l'aide de votre navigateur sur `https://[host]:8443`

Note : Dans le cas d'un serveur Apache déployé sur une autre machine, si vous êtes redirigé vers le serveur Apache et non vers le serveur SmartEA lorsque vous cliquez sur un lien web d'une page SmartEA, alors ajoutez la directive suivante dans votre fichier de configuration Apache :

```
ProxyPreserveHost On
```

5.3.3. Configuration du modeleur SmartEA

Une fois le serveur correctement configuré, côté modeleur, il faut modifier le paramétrage du fichier `application.properties` :

- en vérifiant que le paramètre `webPort` contient bien le numéro de port HTTPS
- en positionnant le paramètre `https=true`

Editez le fichier `application.properties` et modifiez les paramètres suivants :

```
webServer=myhost
webPort=8443
https=true
```

Remplacez `myhost` par le nom d'hôte du serveur ou son adresse IP, **conformément à la façon dont vous avez paramétré votre certificat auto-signé côté serveur**.

Il est également nécessaire de déclarer le certificat au niveau de la JRE utilisée (à l'aide de la commande `keytool -import`, se reporter à la documentation Oracle pour de plus amples informations).

Pour cela, récupérez le certificat (fichier `smartea.cer`) généré sur le serveur.

Importez le ensuite dans le keystore de la JRE qui sera utilisée par le modeleur (le mot de passe du keystore du JDK par défaut étant `changeit`):

```
conf $ keytool -import -alias smartea -trustcacerts -keystore /opt/jdk/jre/lib/security/
cacerts -file smartea.cer -storepass changeit
Owner: CN=myhost, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=Unknown
Issuer: CN=myhost, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=Unknown
Serial number: 3ad97a17
Valid from: Tue Mar 24 11:57:38 CET 2015 until: Mon Jun 22 12:57:38 CEST 2015
Certificate fingerprints:
    MD5:  76:82:F8:4D:20:9C:7F:DD:54:48:CA:84:5C:B5:50:42
    SHA1: 8A:20:F8:A4:7A:E4:ED:F8:5C:4F:D0:E1:21:1D:23:77:8C:F8:15:53
    SHA256:
 38:D0:B6:7A:4D:D0:C1:C3:9D:39:A4:BD:9C:87:57:65:63:00:19:B1:78:4C:AA:CF:CC:43:DF:CE:2A:04:8F:67
    Signature algorithm name: SHA256withRSA
    Version: 3

Extensions:

#1: ObjectId: 2.5.29.17 Criticality=false
```

```
SubjectAlternativeName [
  IPAddress: 192.168.1.10
]

#2: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 73 33 E6 0A 13 D8 62 F3   D5 CB CC 7B F0 0C CC 99  s3....b.........
0010: 8B 02 6F B1                                        ..o.
]
]

Trust this certificate? [no]:  yes
Certificate was added to keystore
```

Remarque : une option de test peut être utilisée, consistant à désactiver les contrôles SSL côté client (`https.trustSelfSignedCertificates=true`), mais celle-ci n'a pas d'effet sur le mécanisme d'installation des plugins côté modeleur qui va donc échouer et ne peut donc pas être utilisée en production (si on l'active il faut s'assurer que le modeleur dispose des plugins de méta-modèle et autres, par exemple dans son répertoire `dropins`).

Démarrez ensuite le modeleur afin de vérifiez qu'il peut se connecter au serveur.

Remarque : si vous avez besoin de remplacer un certificat par un autre, vous devez d'abord supprimer le certificat précédemment installé du keystore :

```
jre/bin/keytool -delete -alias smartea -keystore jre/lib/security/cacerts -storepass changeit
```

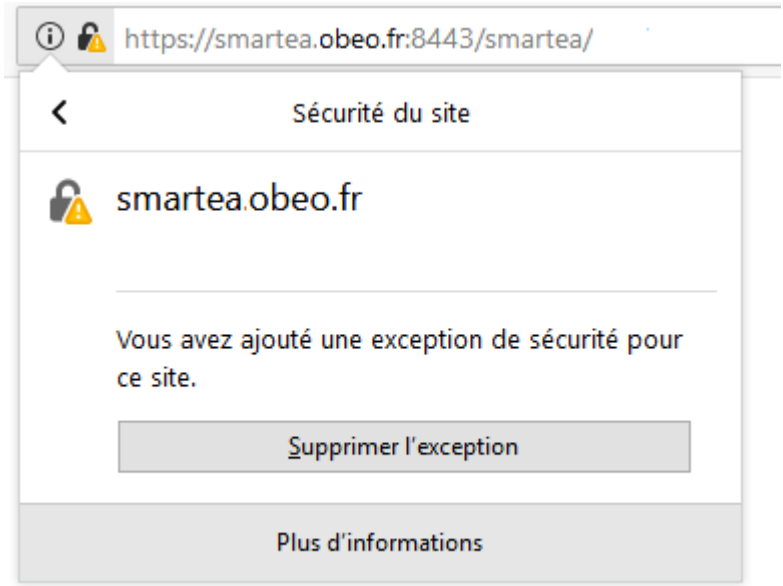
Comment obtenir le certificat à installer sur le modeleur ?

Si vous ne possédez pas le certificat à installer sur votre modeleur, il est possible de le récupérer de la manière suivante :

Ouvrez la page https://[host]:8443 avec le navigateur Firefox.

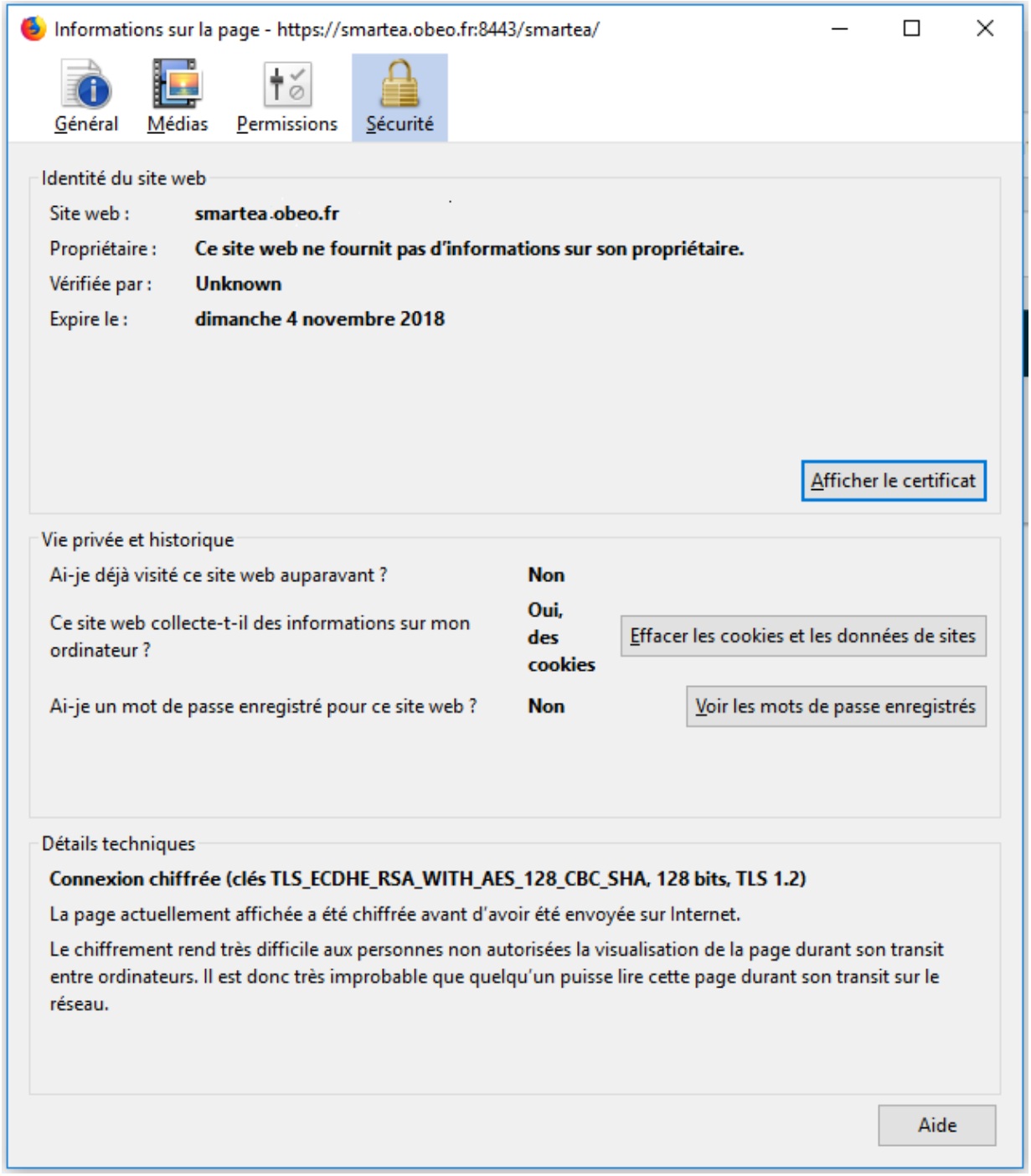
Cliquez sur le cadenas qui se trouve à coté de l'URL.

Affichez les détails de la connexion.

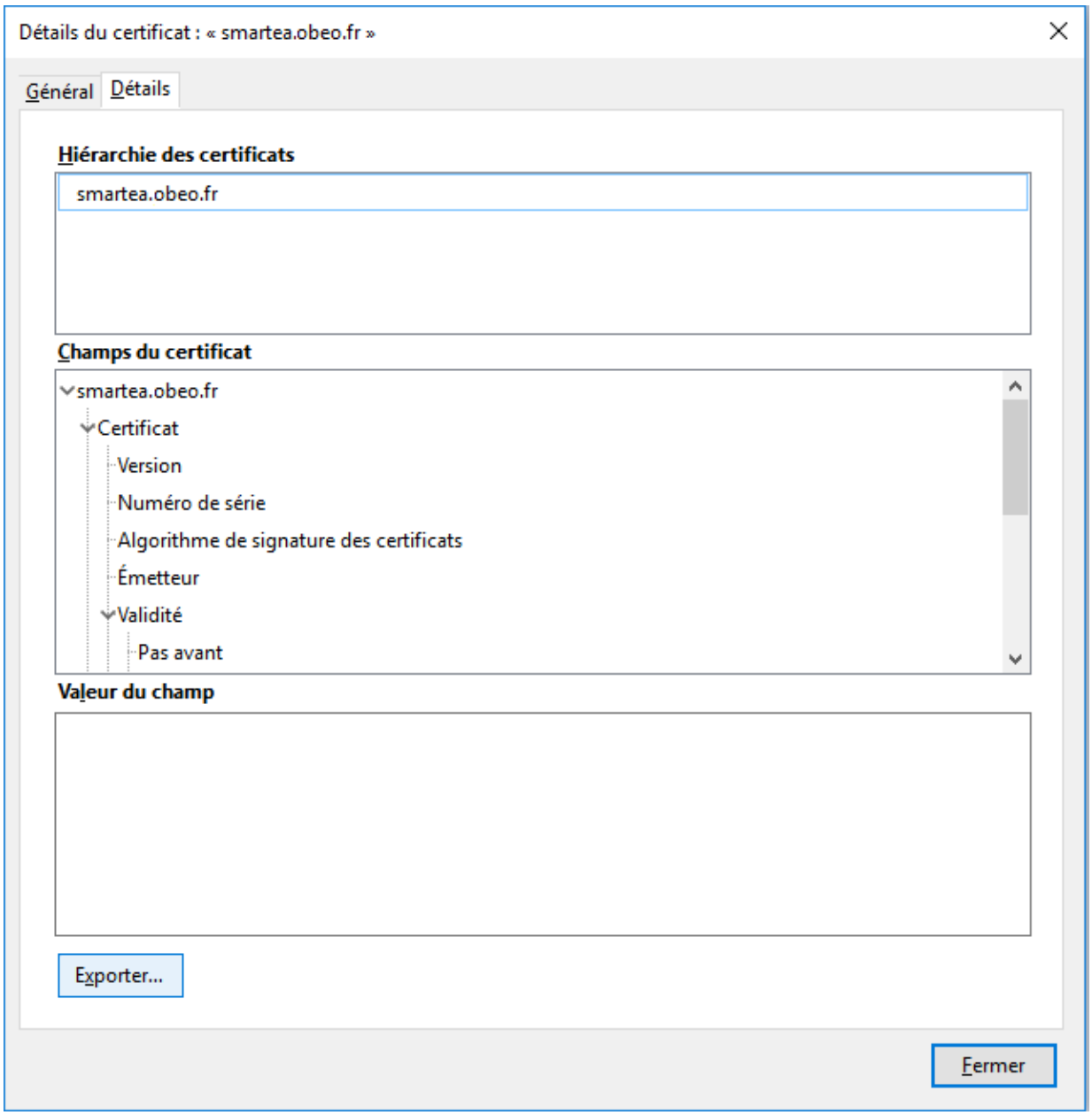


Cliquez sur `Plus d'informations`

Un wizard s'ouvre. Sélectionnez l'onglet `Sécurité`.



Cliquez sur le bouton `Afficher le certificat`.



Exportez ensuite le certificat.

Pour finir, installez le certificat obtenu dans votre modeleur.

5.4. Utilisation d’une connexion SSL entre le serveur CDO et les modeleurs

Il est possible d'activer une connexion SSL entre les modeleurs et le serveur CDO. Les modeleurs et le serveur doivent être configurés en conséquence.

Sur le serveur, un magasin de clés doit être configuré et, du côté des modeleurs, un magasin de clés certifiées contenant la clé publique du magasin de clés du serveur doit être configuré.

5.4.1. Key Store et Trust Store

Keytool peut être utilisé pour créer et gérer des certificats et les magasins. Cet outil est fourni avec le JDK.

5.4.1.1. Générer un magasin de clés

Le magasin de clés contient les informations de certificat, clé privée et clé publique. Pour le générer, utilisez la commande suivante :

```
keytool -genkey -ext SAN=IP:<IP du serveur> -keyalg "RSA" -dname o=sevenSeas -alias
 keystore_alias -keystore server.ks -storepass secret -validity 730 -keysize 4096
```

`-ext` : l'IP du serveur SmartEA.

`-dname` : ce paramètre est facultatif. Il initialise les métadonnées de votre organisation.

Lorsque le mot de passe de la clé est demandé, appuyez sur Entrée pour conserver le même mot de passe que celui du fichier de clés.

5.4.1.2. Signez votre certificat auprès d'une autorité de certification (facultatif)

Cette étape est facultative.

Vous devez transmettre votre demande de signature de certificat (`server.csr`) à votre autorité de certification qui, en retour, fournira un certificat signé (`server.crt`).

```
keytool -certreq -alias keystore_alias -file server.csr -keystore "server.ks"
```

Les deux étapes ci-dessous permettent d'importer un certificat racine et un certificat intermédiaire.

```
keytool -import -alias Root_CA -keystore server.ks -file Root_CA.cer
keytool -import -alias Server_CA -keystore server.ks -file Server_CA.cer
```

Ensuite, importez le certificat signé dans le magasin de clés `server.ks`.

```
keytool -import -alias keystore-signed -keystore server.ks -file server.crt
```

5.4.1.3. Exporter un certificat depuis un magasin de clés

Pour exporter un certificat à partir d'un magasin de clés existant, utilisez la commande suivante :

```
keytool -export -keystore server.ks -alias keystore_alias -file server.cer
```

Cette commande demande la phrase secrète du magasin (Entrez secret conformément à l'exemple), puis crée un fichier `server.cer` contenant le certificat créé précédemment.

5.4.1.4. Créer un fichier de clés certifiées à partir d'un certificat

Il est conseillé de ne pas exporter l'intégralité du fichier de clés sur les modeleurs.

La création d'un fichier de clés certifiées contenant uniquement le certificat et la clé publique est recommandée.

Ce fichier de clés certifiées est destiné à être déployé sur les modeleurs devant se connecter au serveur.

```
keytool -import -file server.cer -alias keystore_alias -keystore trusted.ks -storepass secret
```

Lorsqu'il est demandé «Trust this certificate?», entrez oui.

Cette commande crée un nouveau fichier de clés certifiées dans le fichier `trusted.ks`.

Ce fichier de clés certifiées contient la clé publique du serveur. Il peut être copié sur les ordinateurs clients et référencé via la clé de configuration `truststore.path`.

Le fichier de clés certifiées est protégé par «secret» en tant que phrase secrète.

5.4.2. Configuration du serveur

Activez SSL dans le fichier de configuration `etc/application_cdo.conf` :

```
cdo.protocol=ssl
```

Ajoutez ensuite les lignes suivantes dans le fichier `Obeo-SmartEA-Server.ini` :

```
-Dorg.eclipse.net4j.tcp.ssl.passphrase=secret
-Dorg.eclipse.net4j.tcp.ssl.key=file:///<server.ks absolute path>
```

5.4.3. Configuration des modeleurs

Activez SSL dans le fichier de configuration des modeleurs `application.properties` :

```
cdo.protocol=ssl
```

Ajoutez ensuite les lignes suivantes dans le fichier `Obeo-SmartEA-Modeler.ini` :

```
-Dorg.eclipse.net4j.tcp.ssl.passphrase=secret
-Dorg.eclipse.net4j.tcp.ssl.trust=file:///<trusted.ks absolute path>
```

5.5. Utilisation d'une connexion Web Socket entre le serveur CDO et les modeleurs

Il est possible d'activer une connexion Web Socket (WS) ou Web Socket Secure (WSS) entre les modeleurs et le serveur CDO. Les modeleurs et le serveur doivent être configurés en conséquence.

L'utilisation du protocole Web Socket ou Web Socket Secure permet d'utiliser le port http/https pour la connexion entre le serveur CDO et les modeleurs et non un port dédié.

5.5.1. Configuration du serveur

Activez le protocole Web Socket dans le fichier de configuration `etc/application_cdo.conf` :

```
cdo.protocol=ws
```

ou

```
cdo.protocol=wss
```

Vous devez également renseigner le port utilisé pour la connexion entre le serveur CDO et les modeleurs. Ce port doit être le même que le port utilisé par le serveur Web.

```
cdo.tcpBindingPort=8080
```

Si vous utilisez le protocole `wss`, vous devez configurer Jetty en conséquence : section `sslContextFactory` dans le fichier `etc/jetty-http.xml`.

5.5.2. Configuration du frontal

Si un frontal filtre les accès à votre serveur SmartEA vous devez configurer ce frontal correctement pour rédiriger les requêtes web socket vers le serveur SmartEA.

5.5.2.1. Frontal Apache

Le module Apache mod_proxy_wstunnel doit être activé sur votre serveur. Il fournit le support du tunnelling pour les connexions websocket vers un serveur websockets d'arrière-plan.

Vous devez ensuite ajouter les lignes suivantes dans votre fichier de configuration (le serveur SmartEA écoute sur le port 8080 dans l'exemple ci-dessous) :

```
RewriteEngine on
RewriteCond %{HTTP:Upgrade} websocket [NC]
RewriteCond %{HTTP:Connection} upgrade [NC]
RewriteRule .* "ws://127.0.0.1:8080%{REQUEST_URI}" [P]
```

Ces lignes doivent être ajoutées avant les lignes `ProxyPass` et `ProxyPassReverse`.

Si vous avez activé le protocole wss, vous devez également ajouter avant la ligne `RewriteEngine on` les lignes suivantes :

```
SSLEngine on
SSLCertificateFile /etc/apache2/ssl/<nom du fichier .crt contenant le certificat SSL>
SSLCertificateKeyFile /etc/ssl/private/<nom du fichier de la clé .key>"
```

En dessous de la ligne :

```
Header edit Set-Cookie ^((sessionID|auth)=.*)$ $1;HttpOnly;Secure
```

La ligne suivante doit être ajoutée :

```
RequestHeader edit Cookie ".*auth=\"([a-z0-9]+)\"[^\"]*sessionID=\"([a-z0-9\.]+)\".*" "auth=
\"$1\"; sessionID=\"$2\""
```

Une fois toutes ces modifications faites, noubliez pas de redémarrer le service Apache.

5.5.2.2. Frontal Nginx

Ci-dessous un exemple de modèle de configuration Nginx filtrant les ports 80 (http) et 443 (https) :

```
# Virtual Host configuration for exemple.fr
upstream backend {
   server localhost:8080;
   keepalive 32;
}
server {
    listen 80;
    listen 443 ssl;
    ssl_certificate /path/to/certificate.crt;
    ssl_certificate_key /path/to/certificate.key;
    ssl_protocols       TLSv1 TLSv1.1 TLSv1.2;
    ssl_ciphers         HIGH:!aNULL:!MD5;
    server_name exemple.fr;
    location /smartea/service/subscriptions {
       proxy_set_header Upgrade $http_upgrade;
       proxy_set_header Connection "upgrade";
       client_max_body_size 65M;
       proxy_set_header Host $http_host;
       proxy_set_header X-Real-IP $remote_addr;
       proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
       proxy_set_header X-Forwarded-Proto $scheme;
```

```
        proxy_set_header X-Frame-Options SAMEORIGIN;
        proxy_buffers 256 16k;
        proxy_buffer_size 16k;
        client_body_timeout 60;
        send_timeout 300;
        lingering_timeout 5;
        proxy_connect_timeout 90;
        proxy_send_timeout 300;
        proxy_read_timeout 90s;
        proxy_http_version 1.1;
        proxy_pass http://backend;
    }
    location /smartea/service/net4j {
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        client_max_body_size 65M;
        proxy_set_header Host $http_host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header X-Frame-Options SAMEORIGIN;
        proxy_buffers 256 16k;
        proxy_buffer_size 16k;
        client_body_timeout 60;
        send_timeout 300;
        lingering_timeout 5;
        proxy_connect_timeout 90;
        proxy_send_timeout 300;
        proxy_read_timeout 90s;
        proxy_http_version 1.1;
        proxy_pass http://backend;
    }
    location / {
        proxy_pass http://backend;
    }
}
```

Les paramètres SSL doivent être adaptés en fonction des besoins.

5.5.3. Configuration du modeleur

Activez le protocole Web Socket dans le fichier de configuration des modeleurs `application.properties` :

```
cdo.protocol=ws
```

ou

```
cdo.protocol=wss
```

Si vous utilisez le protocole `wss`, vous devez également renseigner le fichier `Obeo-SmartEA-Modeler.ini`. Ajoutez les lignes suivantes :

```
-Dorg.eclipse.net4j.internal.wss.ssl.passphrase=<mot de passe>
-Dorg.eclipse.net4j.internal.wss.ssl.trust=file:///<certificate.ks absolute path>
```

Important : Si vous avez activé le https et installé le certificat avec l'outil `keytool` dans votre JDK, vous n'avez pas à ajouter ces 2 paramètres. Le certificat est déjà connu.

5.6. Accès au serveur via un proxy HTTP

Il est possible de déployer un modeleur SmartEA sur un poste client où toutes les communications http entre le poste client et le serveur doivent passer par un proxy (par exemple Squid).

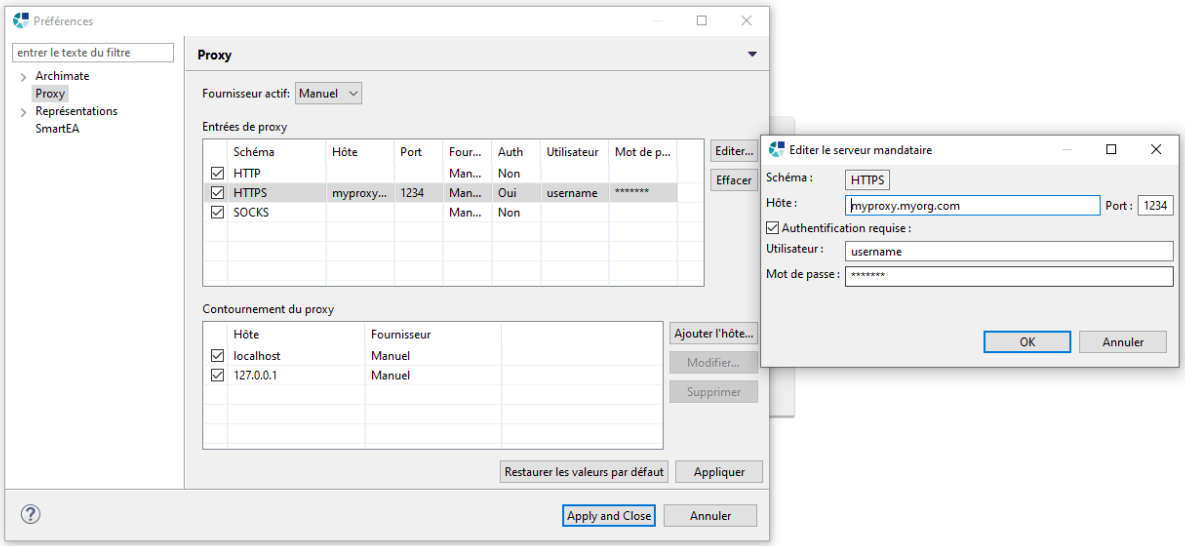
Pour cela il est nécessaire d'utiliser le mode Web Socket Secure (WSS) entre les modeleurs et le serveur comme décrit dans la section précédente.

Une fois cette configuration réalisée et fonctionnelle, vous pouvez activer le mode proxy sur les modeleurs en ajoutant l'option suivante dans le fichier `application.properties` :

```
useProxy=true
```

L'activation de cette option ajoute une nouvelle page de préférences dans les modeleurs permettant de paramétrer un proxy.

Au premier lancement d'un modeleur SmartEA, le proxy n'est pas encore configuré. Si le serveur n'est pas accessible sans passer par le proxy, un dialogue propose l'ouverture de la page de préférences permettant de configurer ce proxy :



Remarque : Si votre proxy nécessite de s'authentifier et selon les paramètres de sécurité de votre poste, il se peut que le client riche ne puisse stocker les informations confidentielles de configuration du proxy (identifiant et mot de passe) dans le répertoire par défaut (sous Windows : `C:\Users\<Compte utilisateur>\.eclipse\org.eclipse.equinox.security\`). Dans ce cas, vous devez modifier cet emplacement. Pour cela, ajoutez dans le fichier `Obeo-SmartEA-Modeler.ini` (juste avant la ligne -vmargs) :

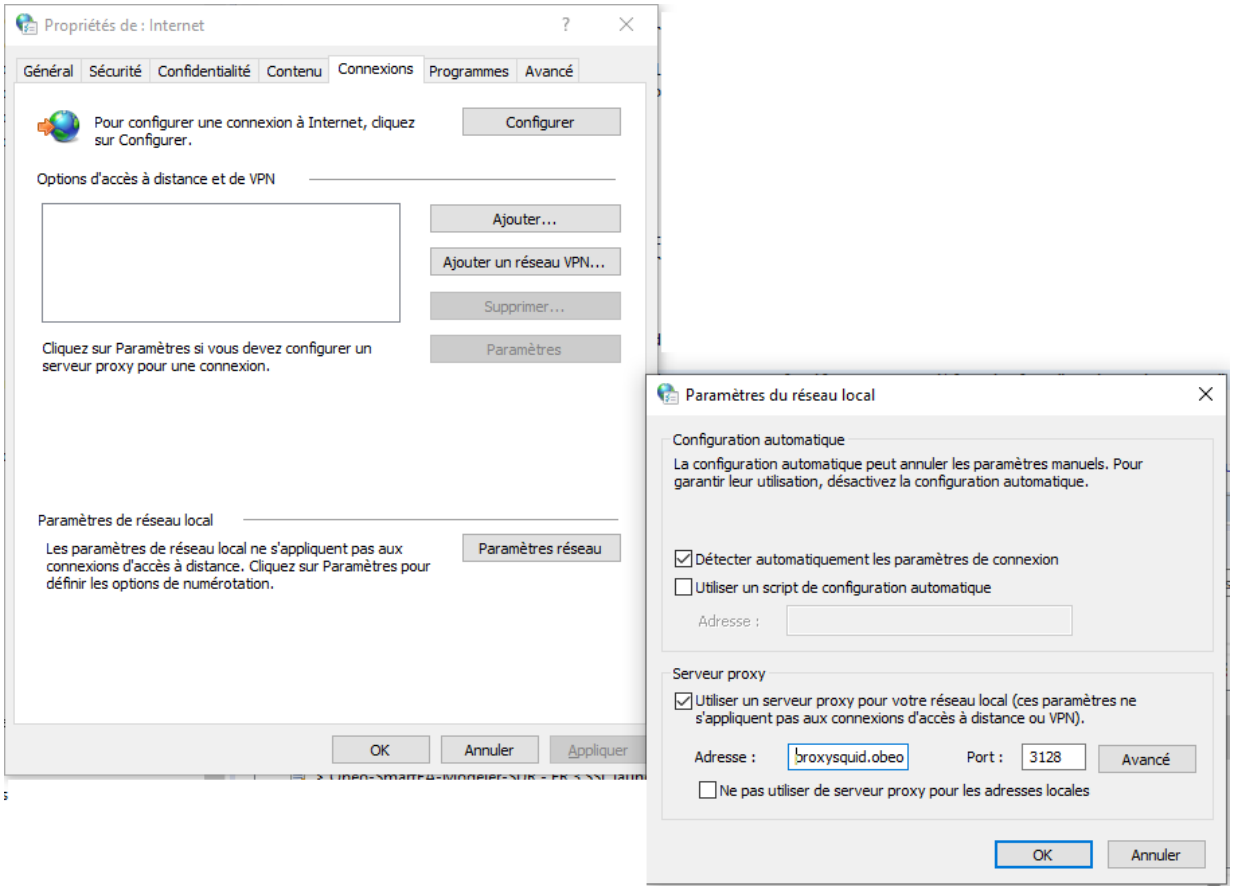
```
-eclipse.keyring
<path>\<nom du fichier secure.storage>
```

Exemple sous Windows :

```
-eclipse.keyring
C:\Users\afontaine\SmartEA\6.2.0\Obeo-SmartEA-Modeler\secure.storage
```

Enfin pour permettre au navigateur web du modeleur d'accéder aux pages il est également nécessaire de paramétrer le proxy au niveau de votre OS.

Ce paramétrage dépend de l'OS, sous Windows par exemple il est accessible via les «Options internet» > Onglet «Connexions» > «Paramètres Réseau».



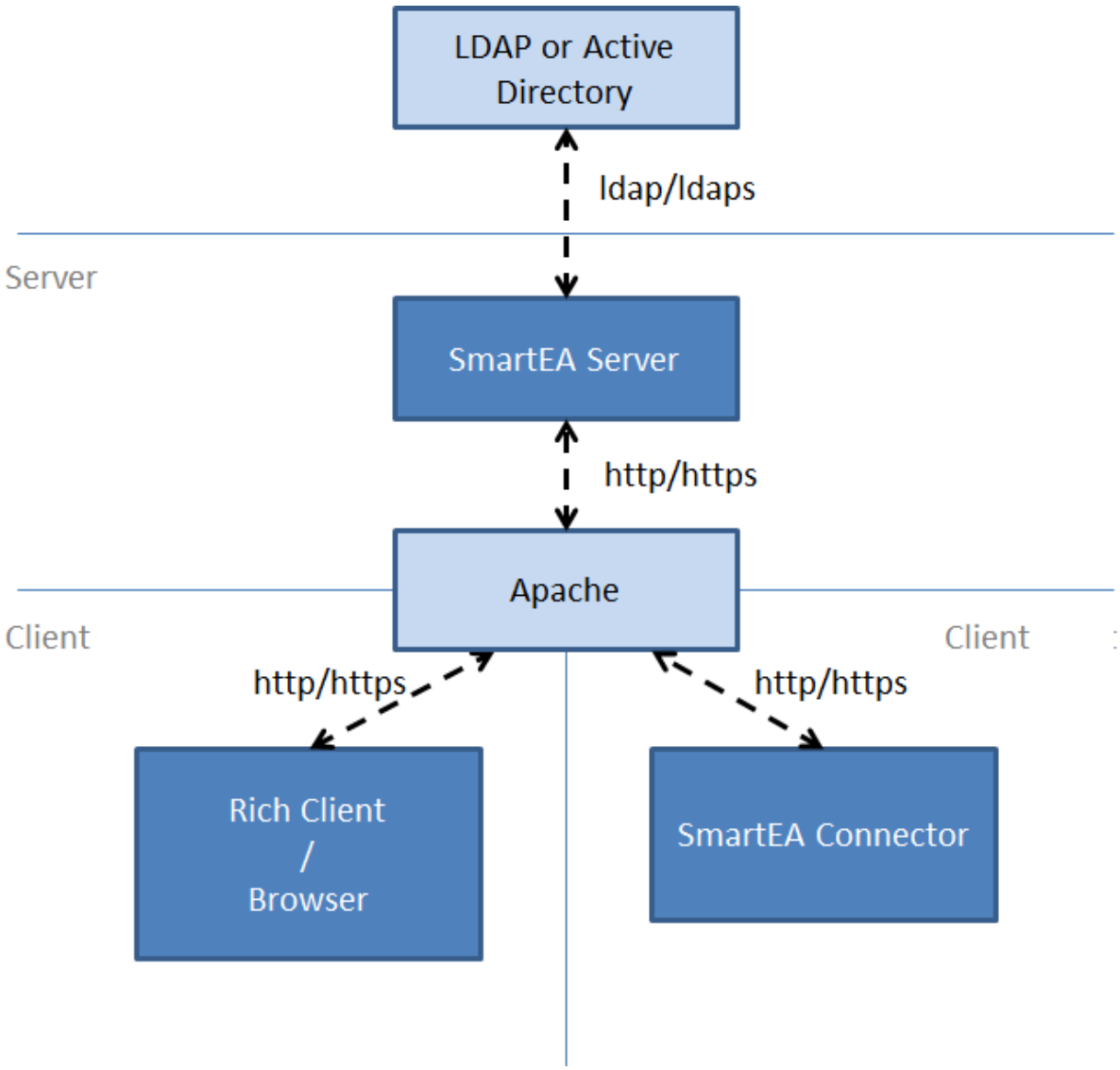
5.7. Authentification

5.7.1. Déploiements possibles

Plusieurs façons d'authentifier les utilisateurs SmartEA sont envisageables. Elles sont listées dans les sous sections suivantes.

5.7.1.1. Utilisation d'un annuaire

Pour connecter SmartEA à l'annuaire d'entreprise, il est nécessaire d'ouvrir le flux `ldap` / `ldaps` entre le serveur SmartEA et l'annuaire.

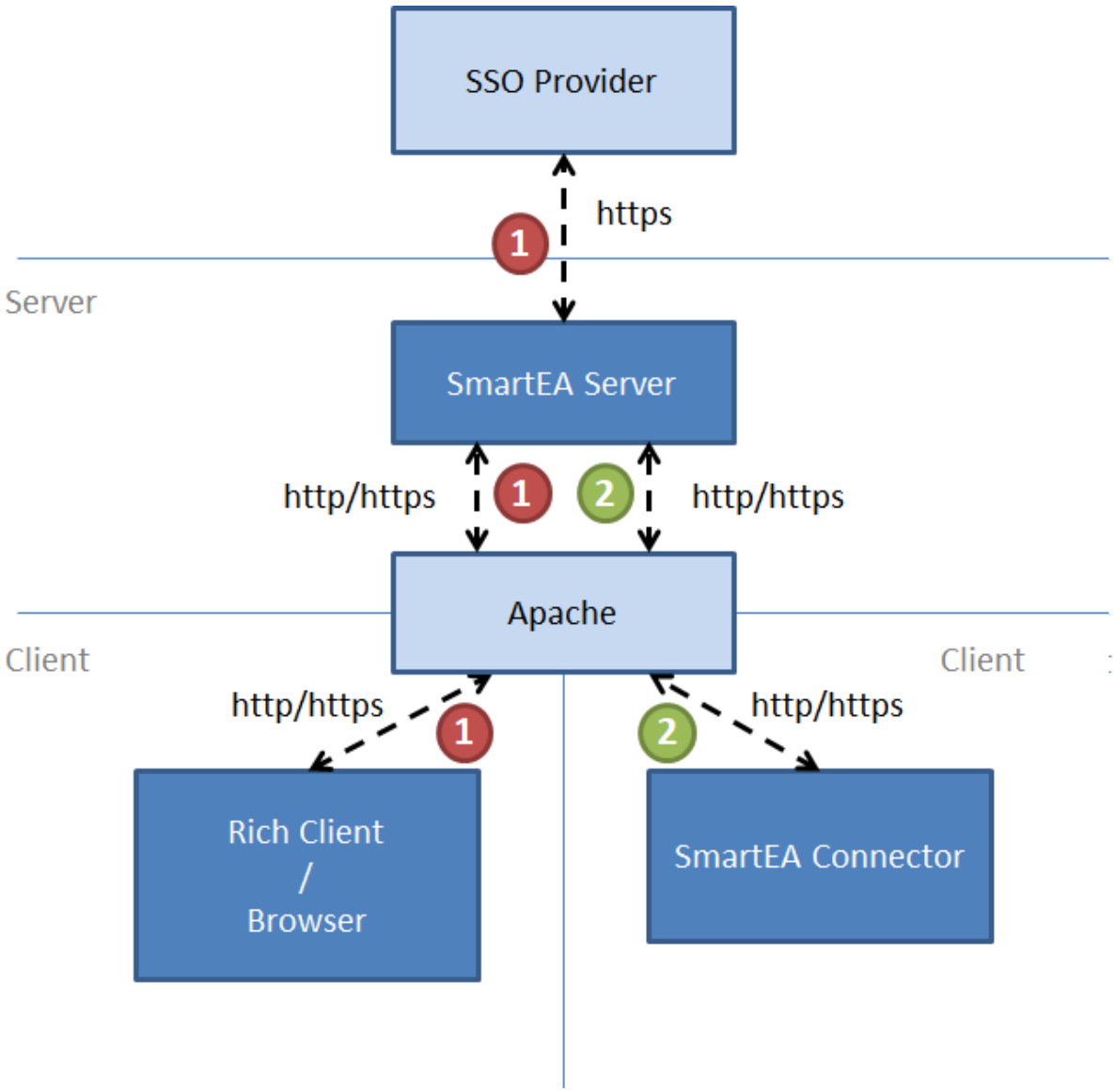


La configuration de SmartEA pour un tel déploiement est décrite en section Configuration de l'authentification LDAP/ Active Directory.

5.7.1.2. Utilisation d'un site tiers SSO et du système d'authentification interne SmartEA pour les comptes de service

Pour connecter SmartEA à un site tiers d'authentification (SSO) et utiliser le système d'authentification interne SmartEA pour les comptes de service, il est nécessaire d'ouvrir le flux `http` / `https` entre le serveur SmartEA et le site d'authentification SSO.

La figure suivante montre le flux d'authentification lorsqu'un utilisateur se connecte par l'intermédiaire d'un client riche ou d'un navigateur web (puce 1) et le flux pour un compte de service (puce 2).



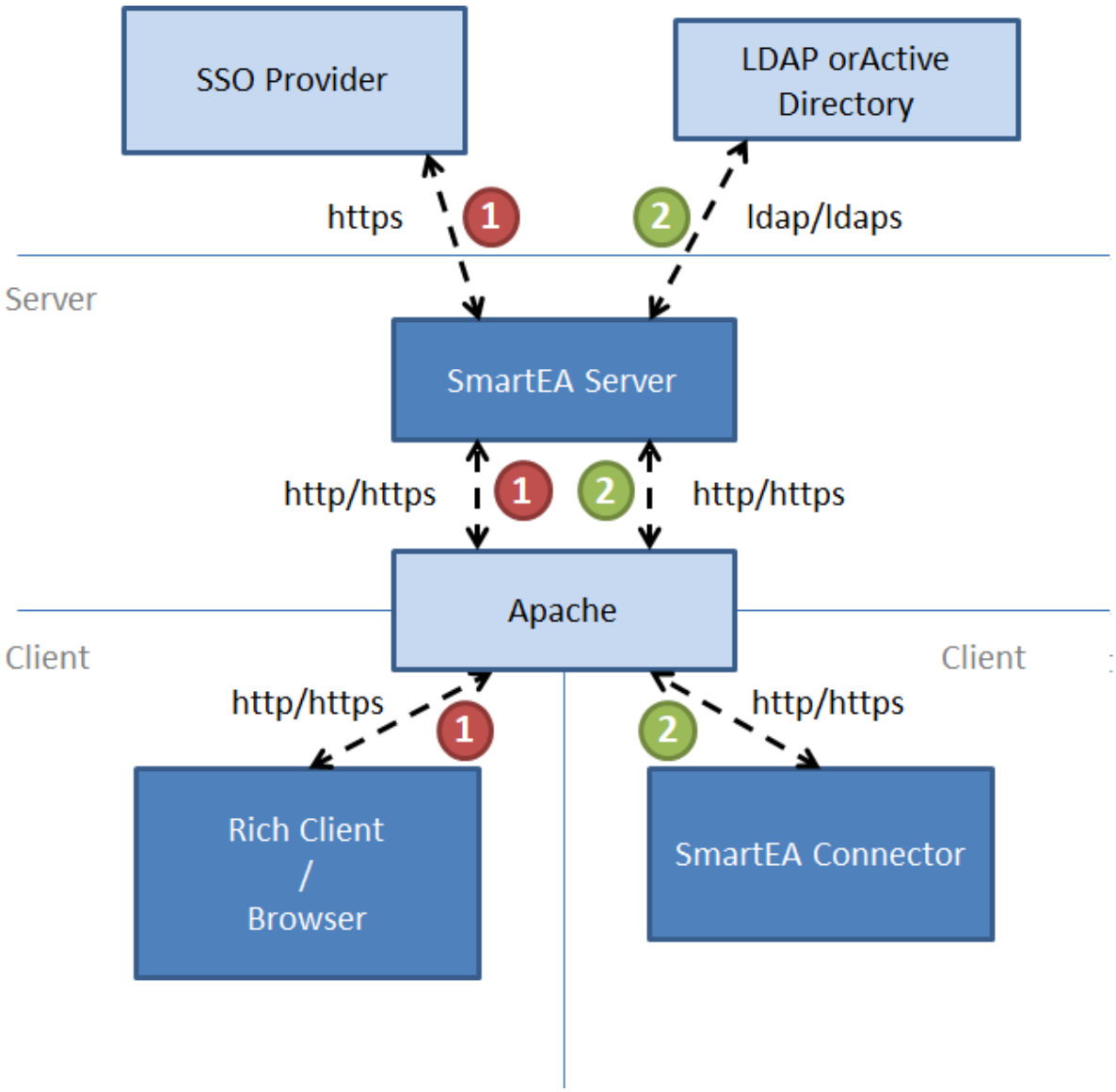
La configuration de SmartEA pour un tel déploiement est décrite en section Configuration de l'authentification SSO.

5.7.1.3. Utilisation d'un site tiers (SSO) et d'un annuaire d'entreprise (LDAP/Active Directory) pour les comptes de service

Pour connecter SmartEA à un site tiers d'authentification (SSO) et à un annuaire d'entreprise (LDAP/Active Directory) pour les comptes de service, il est nécessaire d'ouvrir :

- le flux `http` / `https` entre le serveur SmartEA et le site d'authentification SSO
- le flux `ldap` / `ldaps` entre le serveur SmartEA et l'annuaire.

La figure suivante montre le flux d'authentification lorsqu'un utilisateur se connecte par l'intermédiaire d'un client riche ou d'un navigateur web (puce 1) et le flux pour un compte de service (puce 2).



La configuration de SmartEA pour un tel déploiement est décrite en section Configuration de l'authentification SSO.

5.7.2. Configuration de l'authentification LDAP/Active Directory

Par défaut, le serveur SmartEA gère l'authentification par l'intermédiaire du fichier texte `/etc/users.yml`. En production, il est recommandé de gérer l'authentification par l'intermédiaire d'un annuaire LDAP ou Active Directory.

Pour configurer l'authentification LDAP ou Active Directory, éditez le fichier `etc/application_auth.conf`

Ajoutez une ligne définissant l'implémentation du connecteur LDAP à utiliser. Le serveur SmartEA fournit une implémentation par défaut convenant à la majorité des cas : `fr.obeo.smartea.core.server.impl.user.LdapUserProviderImpl`

```
userprovider.impl=fr.obeo.smartea.core.server.impl.user.LdapUserProviderImpl
```

Précisez ensuite l'URL de l'annuaire :

```
userprovider.ldap.url=ldap://ldap.obeo.fr
```

Attention : Si l'URL commence par ldaps, l'accès au LDAP se fait en ssl. Il est alors nécessaire d'installer un certificat dans le jre utilisé par le serveur.

Si un utilisateur particulier est nécessaire pour se connecter à l'annuaire ajoutez les deux lignes suivantes :

```
userprovider.ldap.appli.dn=le @distinguished name@ (DN) de l'utilisateur applicatif.
userprovider.ldap.appli.password=le mot de passe de l'utilisateur applicatif
```

Précisez ensuite le pattern à utiliser pour récupérer le noeud racine contenant les `distinguisehd names` (DN) des utilisateurs SmartEA :

```
userprovider.ldap.baseDN=dc=obeo,dc=fr.
```

Si le `distinguished name` (DN) de l'utilisateur SmartEA doit être recherché dans le noeud `userprovider.ldap.baseDN` ajoutez la propriété suivante :

```
userprovider.ldap.scope=object
```

Si le `distinguished name` (DN) de l'utilisateur SmartEA doit être recherché dans le noeud `userprovider.ldap.baseDN` et les noeuds fils de premier niveau ajoutez la propriété suivante :

```
userprovider.ldap.scope=oneLevel
```

Si le `distinguished name` (DN) de l'utilisateur SmartEA doit être recherché dans le noeud `userprovider.ldap.baseDN` et tous les noeuds fils quelle que soit la profondeur de l'arbre ajoutez la propriété suivante :

```
userprovider.ldap.scope=subTree
```

Vous pouvez également précisez l'attribut à utiliser pour rechercher le `distinguisehd name` (DN) de l'utilisateur. Par défaut la valeur est `uid`.

Cela convient dans la majorité des cas avec les annuaires LDAP. Si ce n'est pas le cas pour pouvez préciser la valeur de l'attribut.

Dans le cas d'un annuaire Active Directory, il est nécessaire de configurer l'attribut à utiliser. Par défaut cet attribut est `sAMAccountName` :

```
userprovider.ldap.userSearchAttribute=sAMAccountName
```

Vous pouvez ensuite préciser les attributs à utiliser pour retrouver le nom et le prénom d'une personne :

```
userprovider.ldap.firstNameAttribute=...
userprovider.ldap.lastNameAttribute=...
```

Pour finir, vous devez préciser l'implémentation du gestionnaire de profils (`useraccessmanager.impl`) et si nécessaire le chemin du fichier contenant les associations entre les utilisateurs et les prismes (useraccessmanager.yamlbased.userProfilesResource).

Nous proposons une implémentation par défaut du gestionnaire de profils (`fr.obeo.smartea.core.server.impl.user.YAMLBasedUserAccessManagerImpl`). Il permet de relier les utilisateurs d'un annuaire à des prismes. Cela est fait au travers d'un fichier de configuration.`useraccessmanager.yamlbased.userProfilesResource` permet de spécifier le chemin de ce fichier. La valeur par défaut est `etc/profiles.yml`.

```
useraccessmanager.impl=fr.obeo.smartea.core.server.internal.user.YAMLBasedUserAccessManagerImpl
useraccessmanager.yamlbased.userProfilesResource=...
```

Reportez vous à la section présentant le fichier profiles.yml pour plus d'informations.

Nous proposons également une implémentation par défaut du gestionnaire de profils (`fr.obeo.smartea.core.server.impl.user.YAMLBasedUserAndGroupAccessManagerImpl`) permettant de relier les utilisateurs ET les groupes d'un annuaire à des prismes. Si vous utilisez cette implémentation vous devez renseigner les 2 préférences suivantes :

```
userprovider.ldap.group.baseDN=...
```

Ce paramètre permet de définir le pattern à utiliser pour récupérer le noeud racine contenant les groupes, par exemple `ou=Group,dc=obeo,dc=fr`

```
userprovider.ldap.groupSearchPattern=...
```

Ce paramètre permet de définir le pattern pour retrouver un utilisateur particulier dans les différents groupes.

Si le `DN` de l'utilisateur est utilisé dans vos groupes LDAP, utilisez le mot clef réservé `USER_DN` dans votre requête :

- ex LDAP : (&(cn=*)(memberUid=USER_DN))
- ex Active Directory : (&(cn=*)(member=cn=USER_DN))

La variable `USER_DN` est remplacée par le `DN` de l'utilisateur authentifié par le serveur SmartEA.

Si l'identifiant de l'utilisateur est utilisé dans vos groupes LDAP, utilisez le mot clef réservé `LOGIN` dans votre requête :

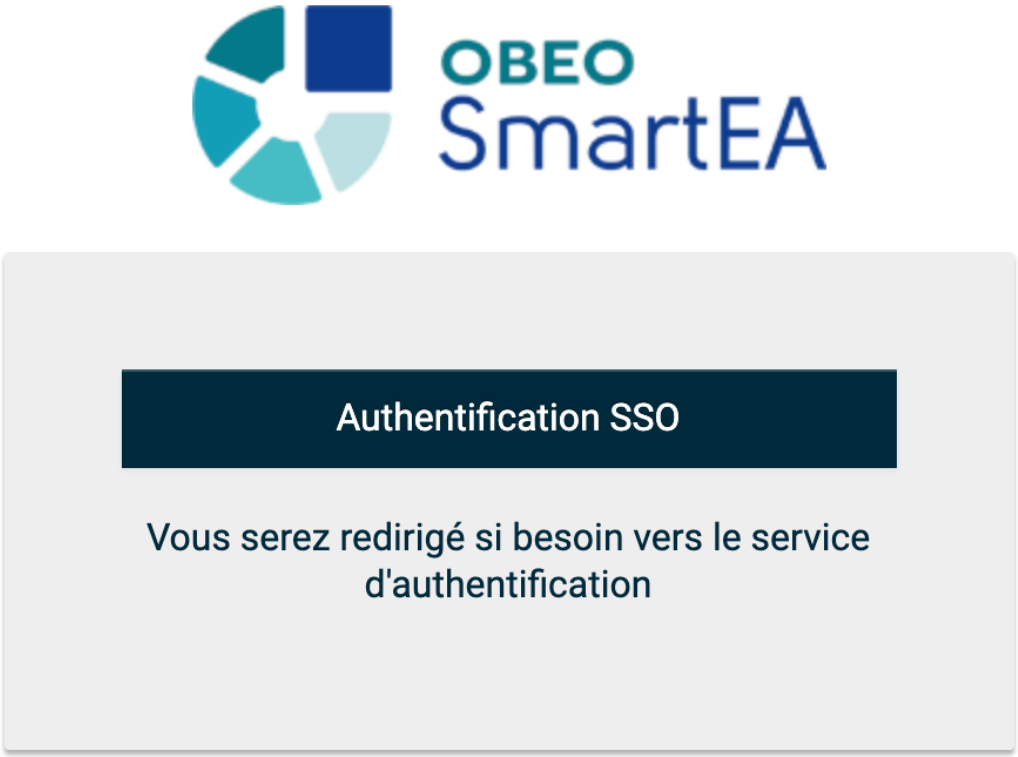
- ex LDAP : (&(cn=*)(memberUid=LOGIN))
- ex Active Directory : (&(cn=*)(member=cn=LOGIN))

La variable `LOGIN` est remplacée par l'identifiant de l'utilisateur par le serveur SmartEA.

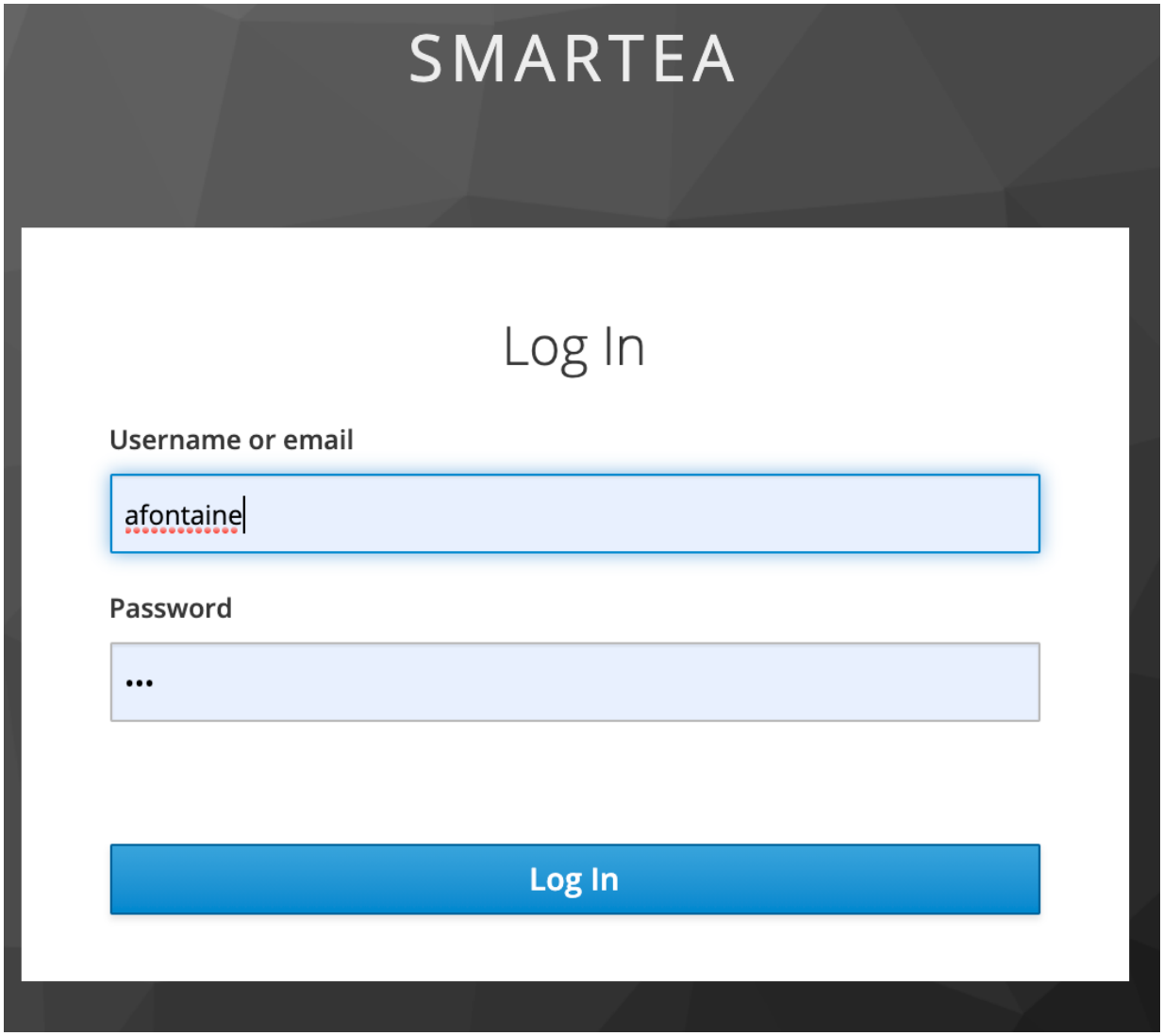
5.7.3. Configuration de l'authentification SSO

L'authentification est aussi disponible en SSO avec le protocole OpenID Connect.

Une fois configuré en SSO, SmartEA affichera à la place du formulaire SmartEA identifiant/mot de passe un lien de redirection vers le site tiers SSO :



En cliquant sur le bouton `Authentification SSO`, vous êtes automatiquement redirigé, si nécessaire, vers le site tiers d’authentification SSO. La capture d'écran suivante montre la page d’authentification de Keycloack.



Dans la suite de cette section, nous détaillons les étapes de configuration. Nous nous appuyons sur l'outil Keycloak (https://www.keycloak.org/) configuré en OpenID afin d'illustrer nos propos.

Ouvrez en édition le fichier `etc/application_auth.conf`

Commencez par ajouter une ligne définissant l'implémentation du connecteur SSO à utiliser. Le serveur SmartEA fournit une implémentation par défaut pour le standard OpenID Connect : `fr.obeo.smartea.core.server.impl.user.OpenIdUserProviderImpl`

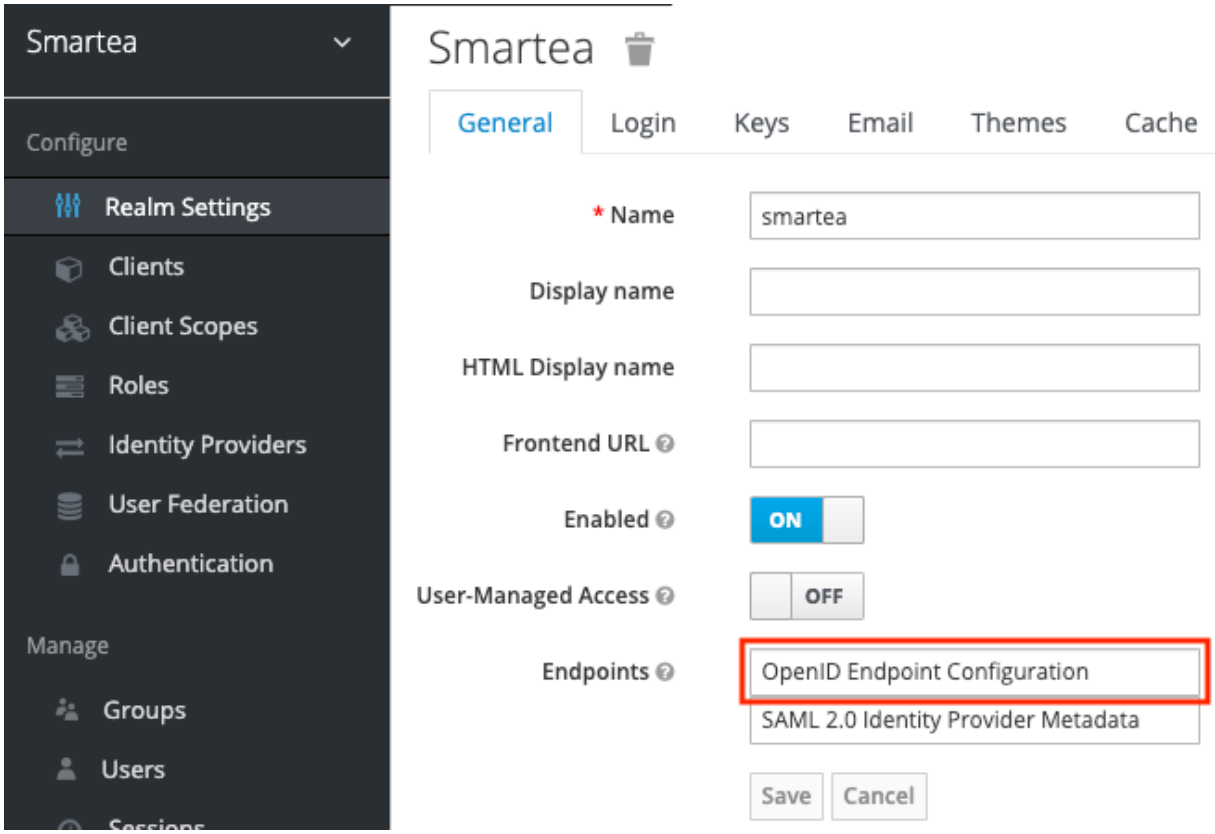
```
userprovider.sso.impl=fr.obeo.smartea.core.server.impl.user.OpenIdUserProviderImpl
```

Attention : l'attribut à utiliser est `userprovider.sso.impl` et non `userprovider.impl` .

Voici un exemple minimal pour paramétrer openID :

```
userprovider.openid.discoveryEndpoint=https://keycloak.obeo.fr:8443/auth/realms/smartea/.well-known/openid-configuration
userprovider.openid.clientId=openid-client
userprovider.openid.clientSecret=d59d8659-2917-4443-be47-8173c2f88410
userprovider.openid.callbackUrl=http://localhost:8080/smartea/sso/callback
```

`userprovider.openid.discoveryEndpoint` permet de découvrir les URLs nécessaires.Dans le «Realm» de Keycloak, on trouve cette URL dans l' `OpenID Endpoint Configuration`.



En cliquant sur la partie entourée en rouge vous obtenez le Json suivant :

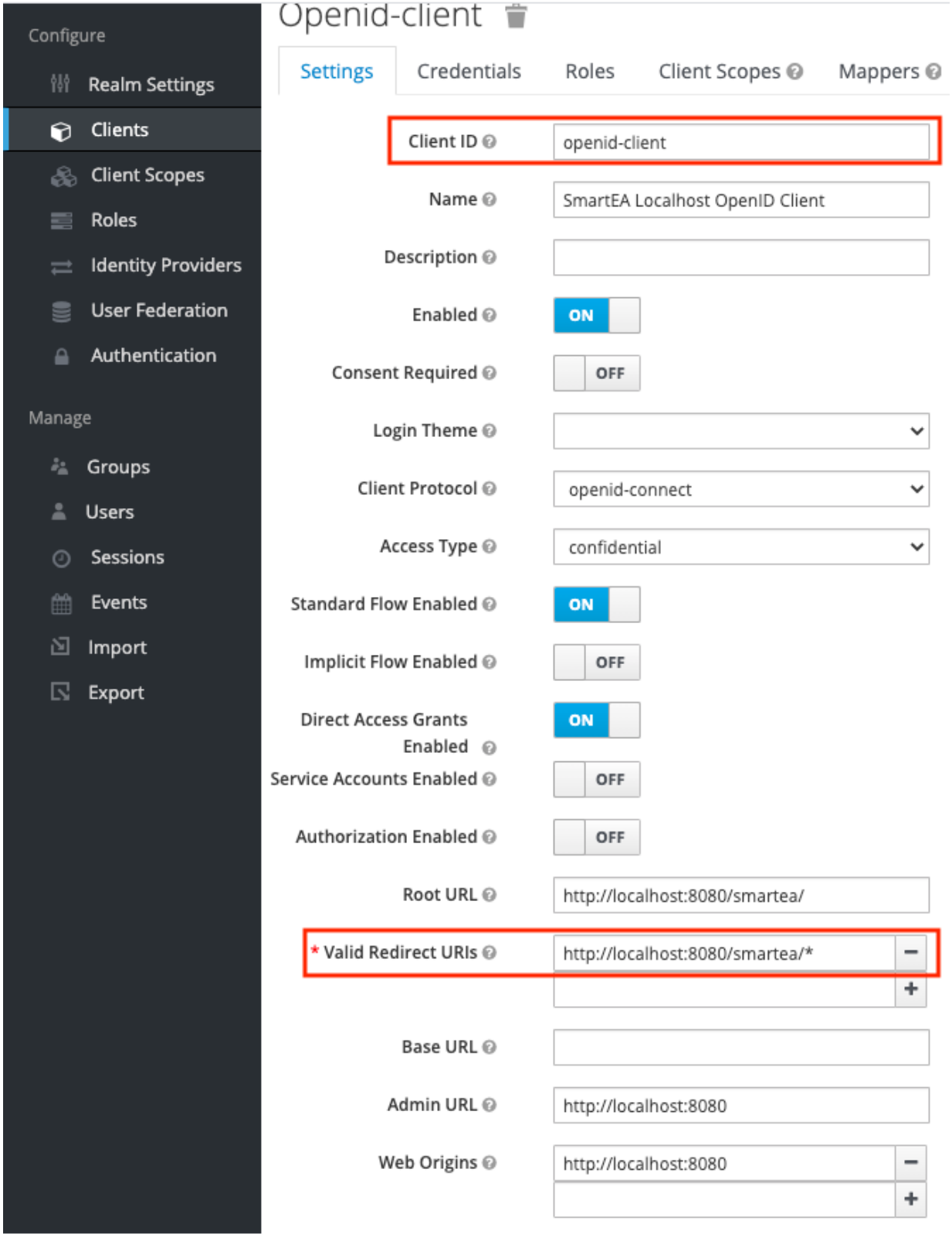
```
{
  "issuer": "https://keycloak.obeo.fr:8443/auth/realms/smartea",
  "authorization_endpoint": "https://keycloak.obeo.fr:8443/auth/realms/smartea/protocol/
openid-connect/auth",
  "token_endpoint": "https://keycloak.obeo.fr:8443/auth/realms/smartea/protocol/openid-
connect/token",
  "token_introspection_endpoint": "https://keycloak.obeo.fr:8443/auth/realms/smartea/protocol/
openid-connect/token/introspect",
  "userinfo_endpoint": "https://keycloak.obeo.fr:8443/auth/realms/smartea/protocol/openid-
connect/userinfo",
  "end_session_endpoint": "https://keycloak.obeo.fr:8443/auth/realms/smartea/protocol/openid-
connect/logout",
  ...
}
```

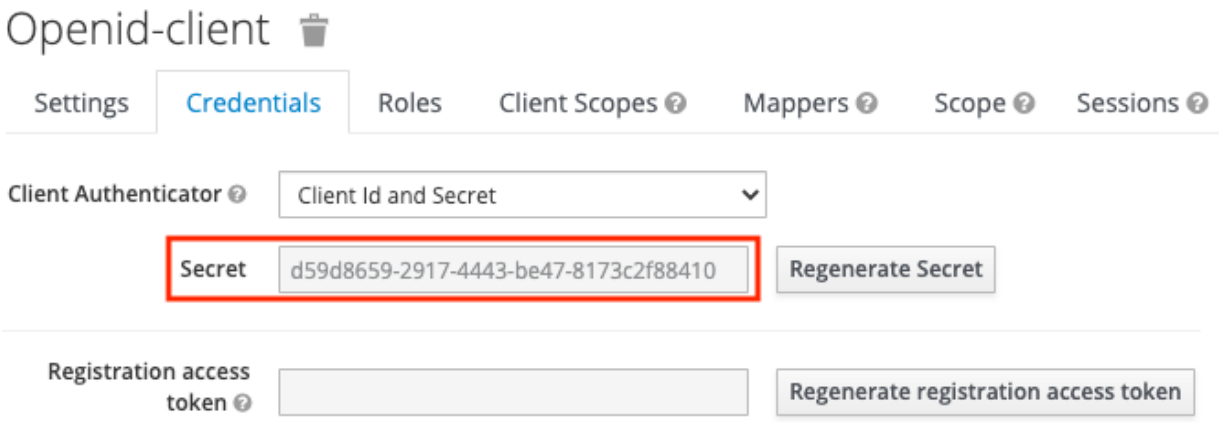
L'administrateur SSO doit également vous fournir les valeurs des attributs `clientId`, `clientSecret` et `callbackUrl`.

Attention : l'url de callback doit absolument suivre le pattern `baseurl/smartea/sso/callback`.

Nous allons maintenant renseigner les attributs `clientId`, `clientSecret` et `callbackUrl` en consultant les informations du client OpenID Keycloak.

L'URL renseignée pour le paramètre `Valid Redirect URIs`, `http://localhost:8080/smartea/sso/callback`, est bien compatible avec `baseurl/smartea/sso/callback`.


Configure
Realm Settings
Clients
Client Scopes
Roles
Identity Providers
User Federation
Authentication
Manage
Groups
Users
Sessions
Events
Import
Export
Openid-client
Settings
Credentials
Roles
Client Scopes
Mappers
Client ID
openid-client
Name
SmartEA Localhost OpenID Client
Description
Enabled
ON
Consent Required
OFF
Login Theme
Client Protocol
openid-connect
Access Type
confidential
Standard Flow Enabled
ON
Implicit Flow Enabled
OFF
Direct Access Grants Enabled
ON
Service Accounts Enabled
OFF
Authorization Enabled
OFF
Root URL
http://localhost:8080/smartea/
* Valid Redirect URIs
http://localhost:8080/smartea/*
Base URL
Admin URL
http://localhost:8080
Web Origins
http://localhost:8080



Nous ajoutons cette configuration :

```
userprovider.openid.clientId=openid-client
userprovider.openid.clientSecret=d59d8659-2917-4443-be47-8173c2f88410
userprovider.openid.callbackUrl=http://localhost:8080/smartea/sso/callback
```

Par défaut, SmartEA :

- Utilise l'expiration `AccessToken`
- Utilise la portée OpenID "openid profile"
- Utilise le OpenID response type "code"
- Obtient l'identité de l'utilisateur à partir des données de l'`IdToken` ou du `userinfo_endpoint` :
  - `preferred_username` pour l'identifiant (à utiliser pour la configuration des profils)
  - `given_name` le prénom à afficher dans Obeo SmartEA
  - `family_name` le nom à afficher dans Obeo SmartEA.
- Ne nécessite pas d'"acrValues"

L'administrateur SSO doit vous confirmer la conformité de ces valeurs avec son service OpenID.Dans le cas contraire, vous pouvez adapter les valeurs `useIdTokenExpiration` / `scope` / `responseType` / `firstNameAttribute` / `lastNameAttribute` / `loginAttribute` / `acrValues` avec la configuration suivante :

```
userprovider.openid.useIdTokenExpiration=true
userprovider.openid.scope=openid profile
userprovider.openid.responseType=code
userprovider.openid.firstNameAttribute=given_name
userprovider.openid.lastNameAttribute=family_name
userprovider.openid.loginAttribute=preferred_username
userprovider.openid.acrValues=1
```

Si l'horloge du serveur SmartEA est décalée avec l'horloge du serveur OpenID, la validation des paramètres temporels ne sera pas correcte.Admettons le contexte suivant :

- serveur SmartEA à 12h00
- serveur OpenID à 12h00 et 10 secondes

Au moment de l'échange et de la validation des jetons OpenID, sans tolérance, SmartEA doit refuser le jeton ayant le paramètre nbf (not before) à 12h00 et 10 secondes.

Ce genre de décalage d'horloge peut se produire pour les serveurs sans accès à internet par exemple.

Par défaut, SmartEA dispose d'une tolérance de 5 secondes.Il est possible de changer cette valeur par configuration :

```
userprovider.openid.leewayWindow=5
```

Pour finir, comme pour la configuration LDAP ou Active Directory, vous devez préciser l'implémentation du gestionnaire de profils (`useraccessmanager.impl`) et si nécessaire le chemin du fichier contenant les associations entre les utilisateurs et les prismes (`useraccessmanager.yamlbased.userProfilesResource`).

Nous proposons une implémentation par défaut du gestionnaire de profils (`fr.obeo.smartea.core.server.impl.user.YAMLBasedUserAccessManagerImpl`). Il permet de relier les utilisateurs à des prismes. Cela est fait au travers d'un fichier de configuration.`useraccessmanager.yamlbased.userProfilesResource` permet de spécifier le chemin de ce fichier. La valeur par défaut est `etc/profiles.yml`.

```
useraccessmanager.impl=fr.obeo.smartea.core.server.internal.user.YAMLBasedUserAccessManagerImpl
useraccessmanager.yamlbased.userProfilesResource=...
```

Reportez vous à la section présentant le fichier profiles.yml pour plus d'informations.

Si vous souhaitez gérer des groupes d'accès il faut ajouter à la configuration le nom de l'attribut contenant les rôles utilisateurs :

```
userprovider.openid.rolesAttribute=roles
```

De plus, si votre serveur OpenID ne renvoie pas les rôles sous formes de simples chaînes de caractères mais d'objets il faut spécifier la clef des objets à utiliser pour obtenir l'identifiant des rôles :

```
userprovider.openid.rolesAsObjects.keyAttribute=
```

Nous proposons une implémentation par défaut du gestionnaire de profils (`fr.obeo.smartea.core.server.impl.user.YAMLBasedSSOUserAndGroupAccessManagerImpl`) permettant de relier les groupes SSO et les utilisateurs à des prismes.

Cycle de vie de la session

Le serveur SmartEA gère le «code flow» pour récupérer les jetons OpenID.Une authentification par «code flow» renvoie au final trois types de jetons et un entier «expires_in» :

- `IdToken` (Obligatoire) la carte d'identité de l'utilisateur au format «JWT»: https://jwt.io/
- `AccessToken` (Obligatoire) un «bearer» pour récupérer des données supplémentaires sur le serveur OpenID dans un format «JWT»: https://jwt.io/ ou opaque
- `RefreshToken` (facultatif) un jetons pour réarmer le `AccessToken`
- `expires_in` (Obligatoire) durée en secondes avant l'expiration du `AccessToken`

Par défaut, SmartEA utilise l'expiration `AccessToken` valorisée par `expires_in`.Lorsqu'une ressource de serveur SmartEA sera récupérée, le serveur testera l'expiration.Si une session SSO devient expirée, SmartEA tentera de réarmer le `AccessToken` grâce au `RefreshToken`.Si le `RefreshToken` est manquant ou si le serveur SSO ne peut pas réarmer le `AccessToken` conformément à sa politique SSO, SmartEA redirigera l'utilisateur vers la page d'authentification.

Vous pouvez configurer SmartEA pour utiliser l'expiration `IdToken` au lieu de l'expiration `AccessToken` (`userprovider.openid.useIdTokenExpiration`).C'est une alternative intéressante si le `IdToken` a une plage horaire plus intéressante ou si le `RefreshToken` est absent dans le serveur SSO.

Validations de jetons

L'`IdToken` a un format « JWT » : https://jwt.io/ et SmartEA valide :

- «alg» l'en-tête algorithme doit être RS256, RS384, RS512, ES256, ES384 ou ES512
- «kid» l'en-tête identifiant certificat utilisé pour forger le jeton. Il doit correspondre à un certificat existant dans le «endpoint» `jwks_uri` présent dans .well-known (Ce «endpoint» est re-récupéré toutes les 6 heures)
- «signature» la signature du jeton correspond bien au certificat
- «iat» le IssuedAt doit être plus ancien que l'heure actuelle
- «nbf» le NotBefore ddoit être plus ancien que l'heure actuelle

- «exp» l'expiration doit être postérieure à l'heure actuelle lorsque la configuration `userprovider.openid.useIdTokenExpiration` est activée
- «acr» doit correspondre à la configuration `userprovider.openid.acrValues`
- «aud» L'Audiences doit contenir au moins le clientId
- «iss» L'Issuer doit correspondre à l'attribut «issuer» du «endpoint» .well-known

Validations de jetons personnalisés

Grâce au SDK développeur SmartEA, vous pouvez contribuer des validations supplémentaires avec le point d'extension nommé `fr.obeo.smartea.core.server.api.openid.tokensValidator`.Vous devez spécifier une implémentation de l'interface API Java `fr.obeo.smartea.core.server.api.openid.ITokensValidatorExtension`.

Comptes de service

Si vous avez besoin de lancer des clients ou des modeleurs de publication (comptes de service) vous devez les déclarer dans le fichier service-users.yml. Reportez vous à la section présentant le fichier pour plus d'informations.

Pour l'authentifcation de ces comptes, deux solutions sont possibles par défaut :

- authentification interne à SmartEA. Dans ce cas vous devez préciser les mots de passe dans le fichier service-users.yml. C'est ce système d'authentification qui est utilisé si vous avez activé le SSO et que vous n'avez pas déclaré de valeur pour l'attribut `userprovider.impl`. Assurez vous qu'aucune valeur n'est associée à ce paramètre.
- authentification grâce à un annuaire LDAP ou Active Directory. Dans ce cas vous devez préciser le connecteur LDAP/Active Directory à utiliser (`userprovider.impl`). Une implémentation par défaut est fournie : `fr.obeo.smartea.core.server.impl.user.LdapServiceUserProviderImpl`. Reporter vous à la section Authentification LDAP/Active Directory pour la configuration.

```
userprovider.impl=fr.obeo.smartea.core.server.impl.user.LdapServiceUserProviderImpl
```

5.8. Configuration de la traçabilité

Des informations de traçabilité peuvent être enregistrées automatiquement à chaque fois qu'un objet sémantique est créé ou modifié.

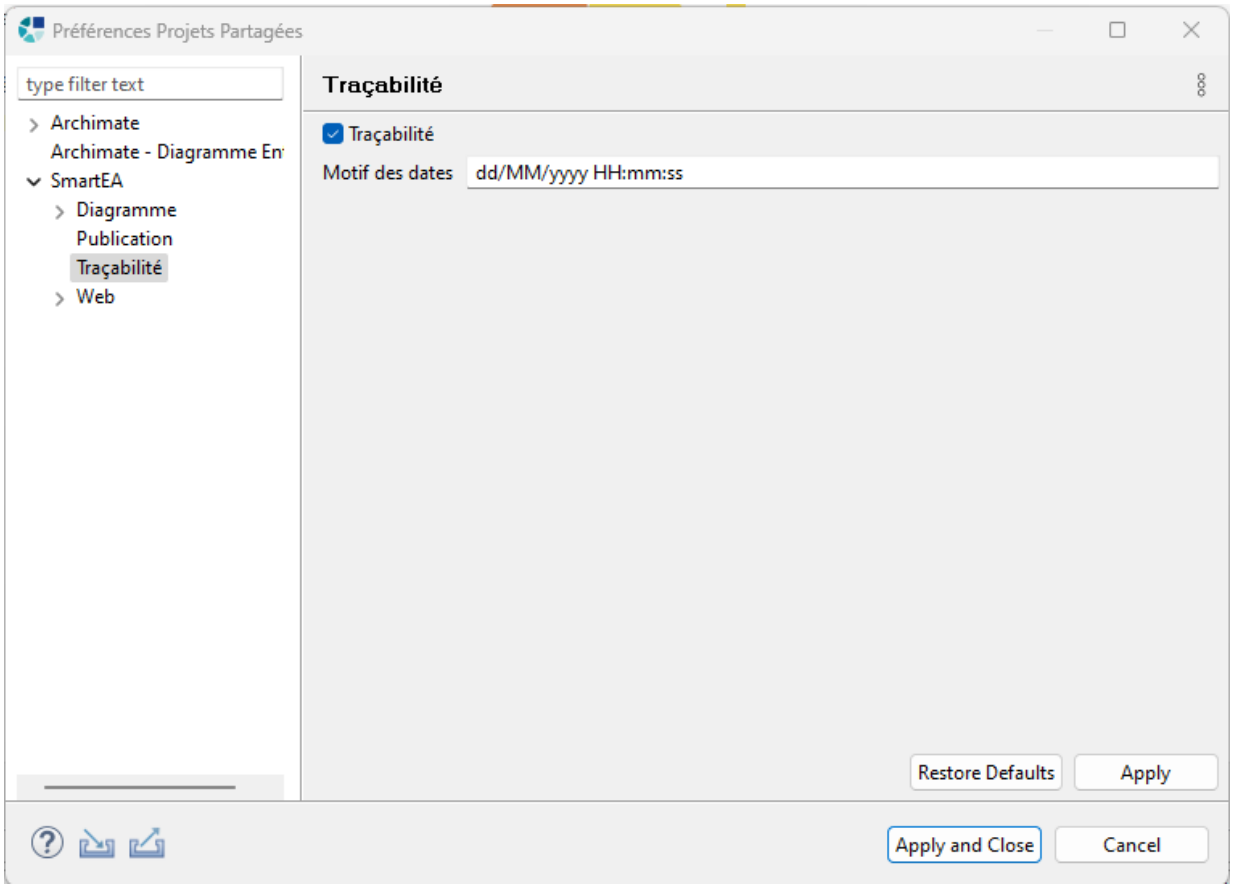
Ces informations sont :

- La date de création de l'objet,
- L'utilisateur ayant créé l'objet,
- La date de dernière modification de l'objet,
- Le dernier utilisateur ayant modifié l'objet.

5.8.1. Configuration

Afin d'activer la Traçabilité sur un projet, ouvrez les Préférences Projet.

Sélectionnez ensuite le noeud SmartEA > Traçabilité.



Cliquez sur la case à cocher **Traçabilité** afin d'activer la traçabilité.

Le champ **Motif des dates** permet de préciser le format d'affichage des dates de création et de modification des objets.

Enregistrez ensuite vos modifications.

Les informations de traçabilité sont maintenant automatiquement enregistrées lors de la création ou modification d'un objet du modèle sémantique : objet sémantique, répertoire ou artefact.

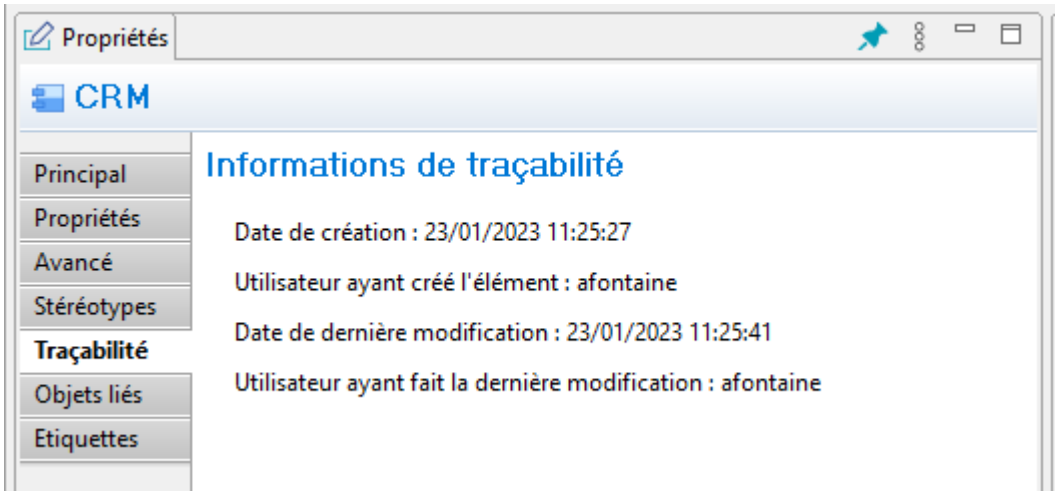
L'enregistrement est fait lors d'une création ou modification provenant du client riche, d'un navigateur web ou d'un connecteur.

Afin que les utilisateurs accèdent à ces informations, il est nécessaire de déclarer la Feature *Traceability* dans chacun des prismes souhaités.

5.8.2. Visualisation des informations de traçabilité

Si dans le prisme courant la Feature *Traceability* est active, alors les utilisateurs auront accès aux informations de traçabilité. Ces informations sont en lecture seule.

Ces informations sont présentées dans le client riche dans l'onglet Traçabilité de la vue Propriétés.



Elles sont également visibles dans l'explorateur de modèle web lorsque l'utilisateur clique sur l'icone Information.

5.9. Utilisation du modeleur comme application de bureau à distance

Le modeleur peut être utilisé de deux manières en mode application de bureau à distance.

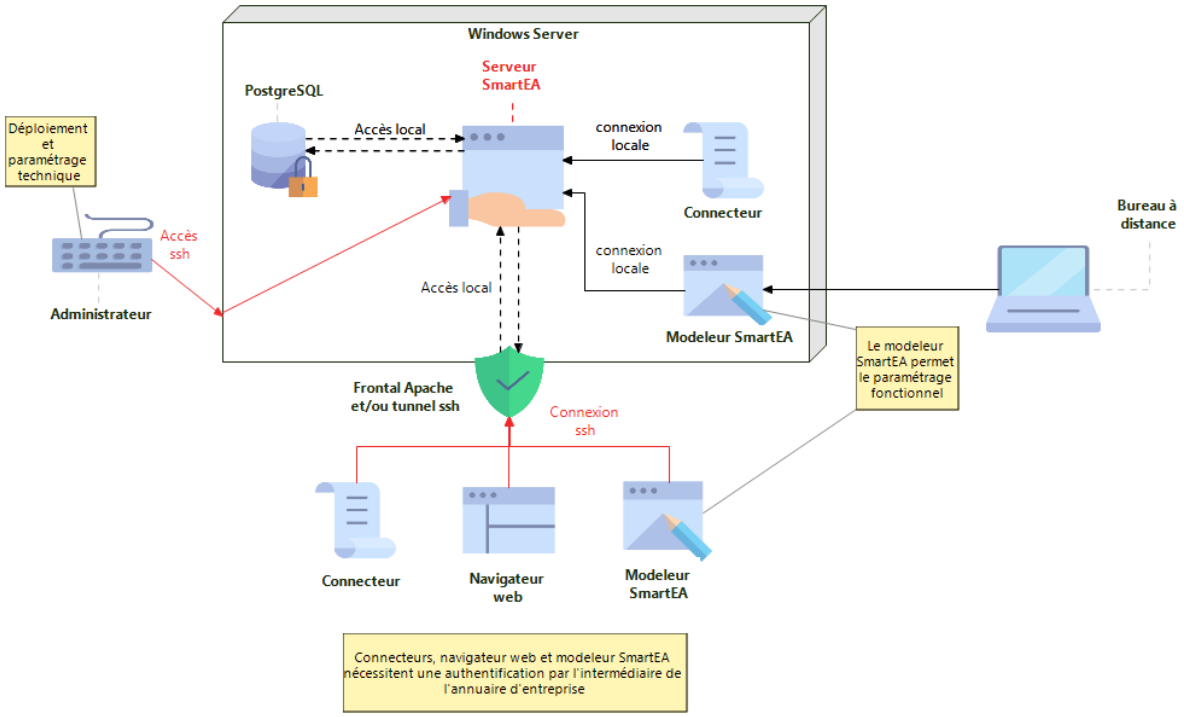
Sur votre serveur de bureaux à distance :

- vous pouvez déployer le modeleur sur chaque compte utilisateur. Dans ce cas, aucune configuration n'est nécessaire.
- vous pouvez partager le modeleur entre chaque compte utilisateur. Dans ce cas, il est nécessaire de configurer le modeleur afin que les utilisateurs aient chacun leurs propres répertoires `workspace` et `configuration`. Pour cela, ajoutez les lignes suivantes dans le fichier `Obeo-SmartEA-Modeler.ini`:

```
-data
@user.home\SmartEA\workspace
-configuration
@user.home\SmartEA\configuration
```

Ces 4 lignes doivent être ajoutées **avant** la ligne `-vmargs` dans le fichier `Obeo-SmartEA-Modeler.ini`.

La figure suivante est un exemple d'architecture de bureau à distance qu'il est possible de mettre en place avec Windows Server.



5.10. Configuration de la publication automatique

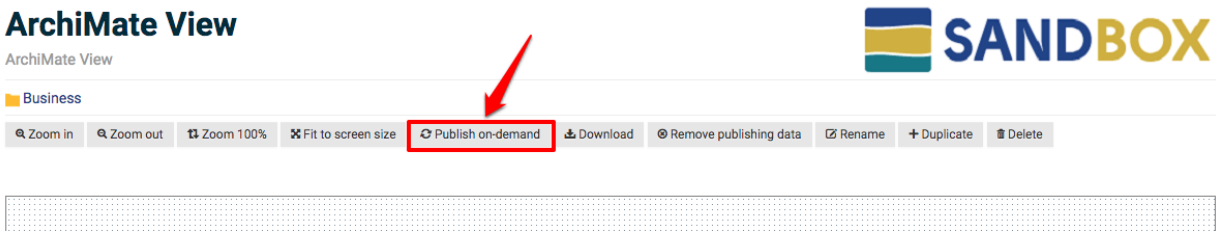
La publication est le processus permettant à SmartEA d'afficher les diagrammes Sirius dans les pages web, sans devoir ouvrir l'éditeur associé dans un modeleur SmartEA.

Ce processus est par défaut automatisé, déclenché à la sauvegarde d'un diagramme mais il est possible de changer ce fonctionnement.

5.10.1. Publication à la demande

Lorsque la publication à la demande est activée, un bouton *Publish on-demand* apparaît sur les pages web des représentations Sirius.Ce bouton permet aux utilisateurs de demander la publication de la représentation Sirius depuis un navigateur web.

La publication à la demande est activée depuis l'éditeur de prismes. Pour cela ajoutez la feature *OnDemandPublishing*.



5.10.2. Publication sur commit

La publication sur commit permet de déclencher la publication d'un ensemble de représentations Sirius suite à un commit sémantique.

La publication sur commit se configure dans un fichier XML qui se trouve sur le serveur dans le répertoire `etc/projects`.Ce fichier doit être nommé « id_projet ».publication. Cette configuration est donc spécifique à un projet.

Ci-dessous le format du fichier xml de configuration :

```
<publications>
    <on-commit-publications>
       <!-- Publier la représentation _rT43257sHgvy sur toutes les branches -->
       <include branchId="*" logicalId="_rT43257sHgvy" />
       ...
    </on-commit-publications>
</publication>
```

- `branchId` : paramètre permettant de préciser une branche particulière (id de la branche) ou toutes les branches (« * »).
- `logicalId` : identifiant logique de l'artefact Sirius devant être publié suite à un commit. Ce paramètre est obligatoire.

Remarque : une ligne « include » est nécessaire pour chaque représentation à publier suite à un commit.

5.10.3. Publication massive

La publication massive permet de déclencher la publication d'un ensemble de représentations Sirius par une action externe à SmartEA (appel d'un service REST).

La publication massive se configure dans le même fichier de configuration que celui de la publication sur commit.Cette configuration est donc spécifique à un projet.

Ci-dessous le format du fichier xml de configuration :

```
<publications>
    <massive-publications>
       <!-- Publier l'ensemble des artefacts, de la branche master, du viewpoint
 EnterpriseArchitecure et de la représentation ApplicationPortfolio -->
       <include branchId="Master" viewpointId="EnterpriseArchitecture"
 representationId="ApplicationPortfolio" logicalId="*" />
       <!-- Publier l'ensemble des artefacts de la branche Master -->
       <include branchId="Master" viewpointId="*" representationId="*" logicalId="*" />
       <!-- Ne pas publier la représentation _Rxcb44T43257 -->
       <exclude branchId="*" viewpointId="*" representationId="*"  logicalId="_Rxcb44T43257" /
>
    </massive-publications>
</publications>
```

- `branchId` : paramètre permettant de préciser une branche particulière (id de la branche) ou toutes les branches (« * »).
- `viewpointId` : paramètre permettant de préciser un identifiant de viewpoint Sirius en particulier ou tous les viewpoints Sirius (« * »).

- `representationId` : paramètre permettant de préciser un identifiant de représentation Sirius en particulier ou toutes les représentations Sirius (« * »).
- `logicalId` : Identifiant logique de l'artefact Sirius devant être publié.

5.10.4. Modeleur de publication

Il est nécessaire de démarrer au moins un modeleur de publication pour traiter les demandes de publication (massive, à la demande ou sur commit).

Pour cela, démarrer le modeleur à partir d'un terminal ou à l'aide d'un raccourci (sous Windows) avec les paramètres suivants :

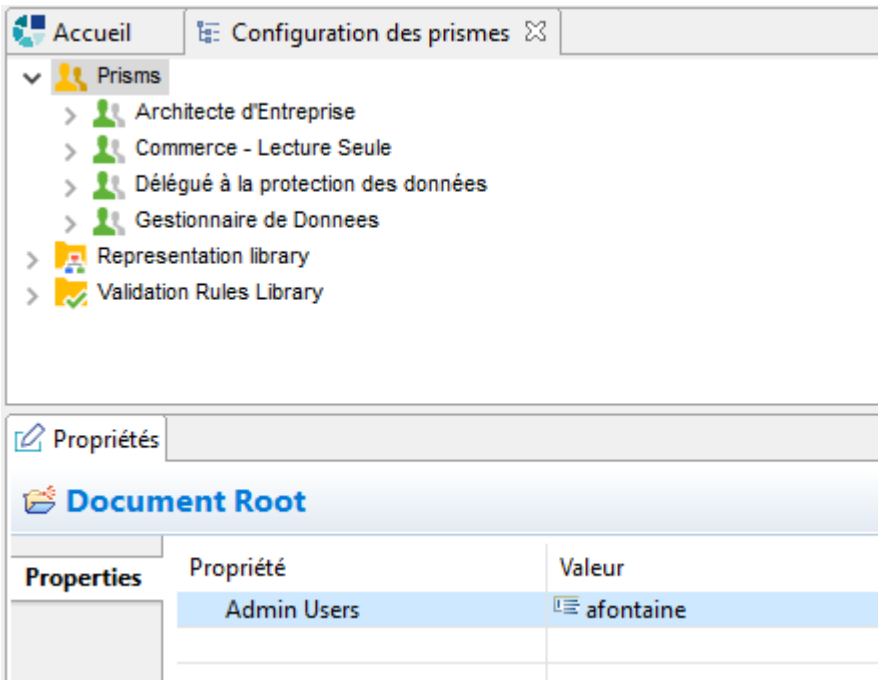
```
-publisher -u [user] -p [user password] -project [project name] -prism [prism id] -branch
 [branch name]
```

Le dernier paramètre est optionnel. Il permet de limiter l'action d'un modeleur de publication à une branche précise.

Dans ce mode, le modeleur reste actif et visible en attente des traitements à effectuer.

Remarque : Un modeleur de publication ne peut que traiter les demandes de publication.

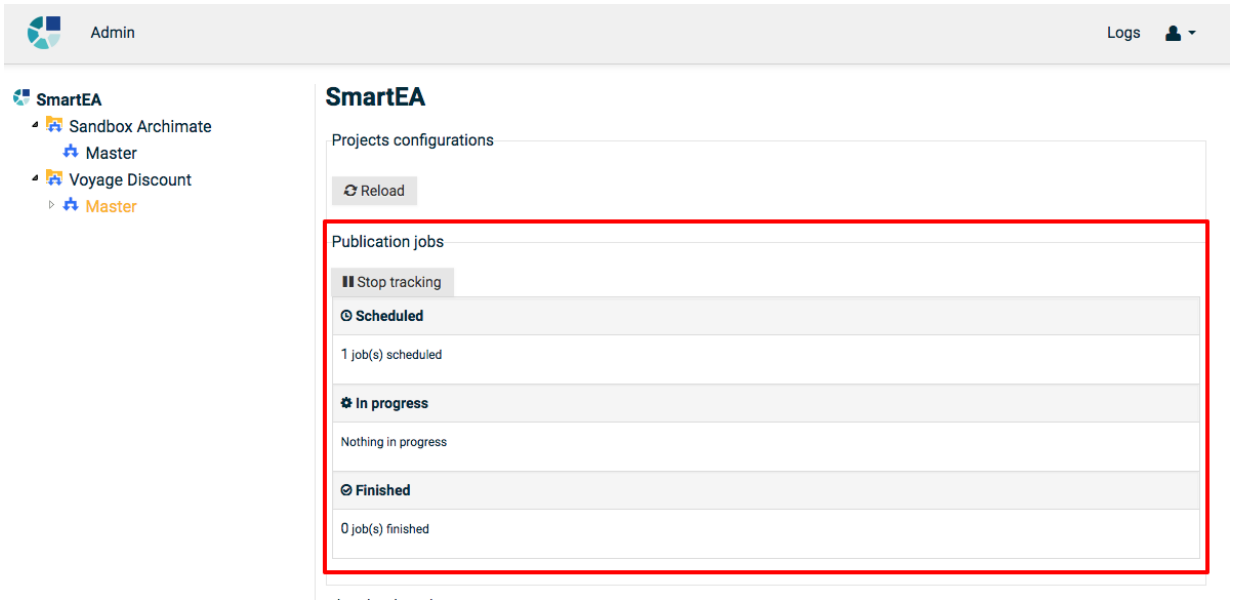
Attention : L'utilisateur utilisé par le modeleur de publication doit être déclaré comme administrateur.



5.10.5. Administration de la publication

Suivi des demandes de publication

Que ce soit à la demande, sur commit ou massive, lorsqu'une publication est demandée, des travaux de publication sont créés.Il est possible de voir la liste des travaux en attente, en cours de traitement et terminés sur la page d'administration de SmartEA.



API REST

Quelques opérations accessibles via une API REST permettent une gestion des travaux :

- **/jobs/logs** – Retourne un fichier de log faisant état des travaux.
- **/jobs/purge/all** – Supprime tous les travaux, qu'ils soient en attente, en cours de traitement ou finis.
- **/jobs/purge/finished** – Supprime tous les travaux terminés.

5.10.6. Automatisation de la publication massive

Que ce soit sur Linux ou Windows, il est possible de lancer automatiquement et périodiquement la publication massive.

Sous Linux, cela peut être fait par exemple par l'ajout d'une nouvelle entrée dans la crontab.Sous Windows, cela peut être fait par exemple à l'aide du planificateur de tâches.

Linux : Crontab

Pour définir une tâche automatique sous Linux, il est nécessaire d'ajouter une entrée à la crontab. Pour cela éditez le fichier /etc/crontab en mode administrateur afin d'ajouter une ligne ayant la forme suivante :`* * * * * user command`

Cette entrée dans la crontab se caractérise par plusieurs paramètres :

1. minute (0-59)
2. heure (0-23)
3. jour dans le mois (1-31)
4. numéro du mois (1-12) ou l'abréviation du mois (jan, fev, …)
5. nom ou chiffre du jour de la semaine (0 représentant dimanche)
6. utilisateur utilisé pour lancer la commande
7. commande à exécuter

Pour chaque unité de temps, les notations suivantes sont possibles :

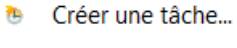
- * : à chaque unité de temps
- 2-5 : intervalle d'unités de temps (2,3,4,5)
- */3 : toutes les 3 unités de temps (0,3,6,…)
- 5,8 : les unités de temps 5 et 8

Ci-dessous, voici un exemple permettant de lancer automatiquement une tâche de publication massive tous les jours à 3H avec l'utilisateur root :`0 3 * * * root curl –user [user]:[password] http://[host]:[port]/smartea/[project_id]/[branch_id]/service/publication/massive`

Les paramètres [user], [password], [host], [port], [project_id] et [branch_id] doivent être modifiés pour correspondre à votre configuration.

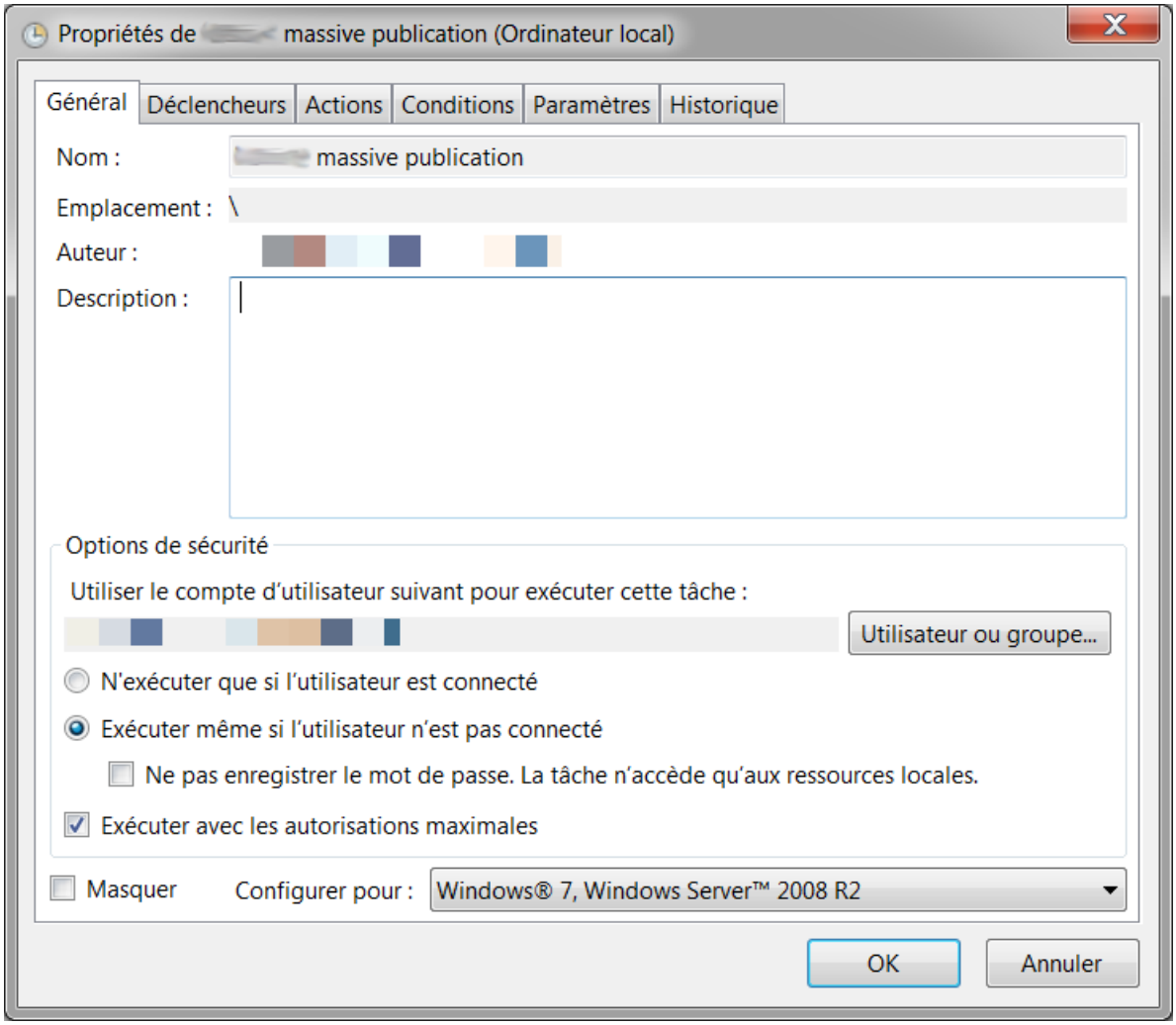
Windows : Planificateur de Tâches

Pour définir une tâche automatique sous Windows il est possible d'utiliser le planificateur de tâches.

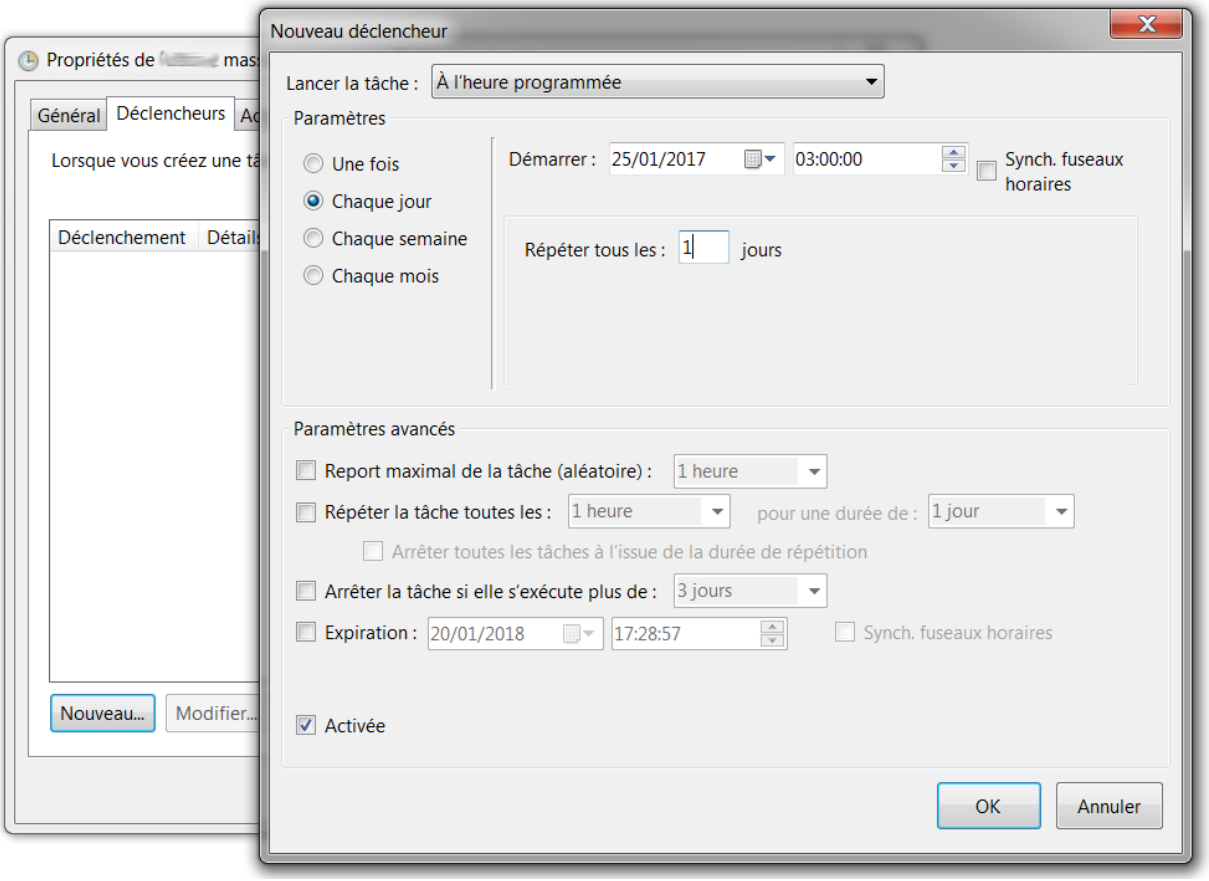
À partir du planificateur de tâches cliquez sur : 

L'exemple suivant explique comment lancer l'exécution automatique d'un script powershell (version 3 minimum) tous les jours à 3H00.

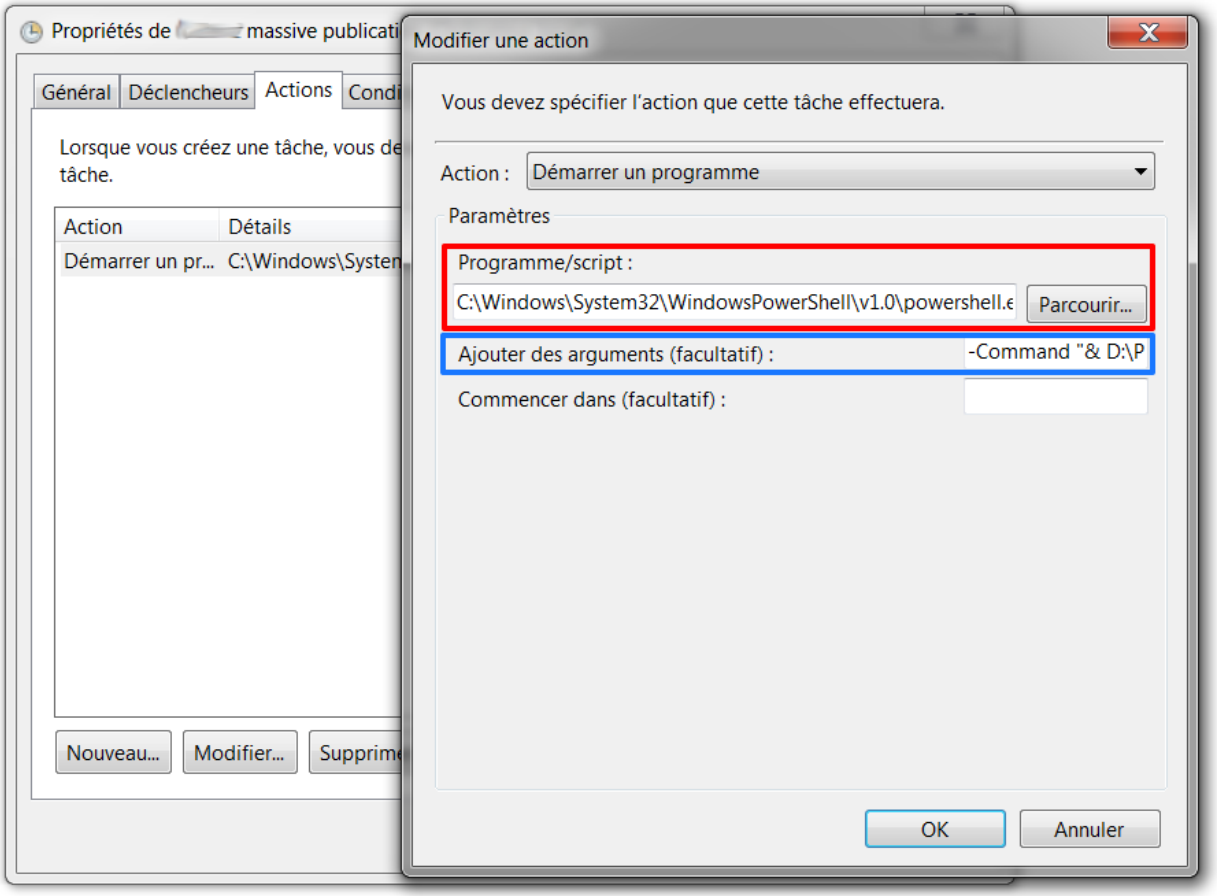
- Onglet général :



- Onglet Déclencheur -> Cliquez sur nouveau :



- Onglet Action -> Cliquez sur nouveau :



Dans le champ texte associé à « Programme/script » (encadré rouge), entrez le chemin vers powershell. C:\Windows \System32\WindowsPowerShell\v1.0\powershell.exe dans l'exemple.

Dans le champ texte associé à « Pour Ajouter des arguments » (encadré bleu), saisissez les arguments suivants :

```
-Command "& Path\To\script\powershell\Launch_massive_publication.ps1 -utilisateur [user] -mot_de_passe [password] -hote [host] -port [port] -projectId [projectId] -branchId [branchId]"
```

Les paramètres [user], [password], [host], [port], [project_id] et [branch_id] doivent être modifiés pour correspondre à votre contexte.

Le script `Launch_massive_publication.ps1` est donné en fin de section.

- Onglets Conditions et Paramètres :

Vérifiez que les options de ces deux onglets sont correctes puis validez la création de la tâche planifiée.

Script de la publication massive

```
Param(
[parameter(Mandatory=$true)][String]$utilisateur,
[parameter(Mandatory=$true)][String]$mot_de_passe,
[parameter(Mandatory=$true)][String]$hote,
[parameter(Mandatory=$true)][String]$port,
[parameter(Mandatory=$true)][String]$projectId,
[parameter(Mandatory=$true)][String]$branchId
)
$params = @{
   uri = 'http://' + $hote + ':' + $port + '/smartea/' + $projectId + '/' + $branchId + '/service/publication/massive';
   Method = 'Get';
   Headers = @{
```

```
      Authorization = 'Basic ' +
 [Convert]::ToBase64String([Text.Encoding]::ASCII.GetBytes("$($utilisateur):
$($mot_de_passe)"));
   } #end headers hash table
} #end $params hash table
$var = Invoke-WebRequest @params
```

Remarque : par défaut, l'exécution de scripts powershell peut être désactivée sur le système. Pour lever cette restriction pour l'utilisateur courant, tapez la commande suivante dans powershell :`Set-ExecutionPolicy -Scope "CurrentUser" -ExecutionPolicy "Unrestricted"`

5.11. Génération basée sur des templates en ligne de commande

La génération de diagrammes basée sur des templates peut être lancée en ligne de commande.

Les diagrammes sont générés grâce à un modeleur démarré avec un paramètre particulier ne necessitant aucune intervention utilisateur : `-templategenerator`. Aucune fonctionnalité n'est accessible aux utilisateurs lorsqu'un modeleur est démarré avec ce paramètre.

Dans ce mode, le modeleur effectue les mises à jour demandées et s'arrête en renvoyant un code de retour.

Ce mode a besoin :

- d'une interface graphique. Une interface graphique doit donc être installée sur la machine executant cette fonctionnalité.
- que Java soit installé sur la machine (même version que celle utilisée pour les modeleurs).

5.11.1. Configuration

Le modeleur qui sert à la génération automatique de diagrammes doit être correctement paramétré. Pour cela vous devez :

- renseigner les bonnes valeurs dans le fichier `application.properties`,
- renseigner les bonnes valeurs dans le fichier `Obeo-SmartEA-Modeler.ini` (paramètre Xmx notamment),
- paramétrer les informations concernant le proxy HTTP si nécessaire,
- mettre à jour les plugins du modeleur.

Note : Afin de s'assurer du bon paramétrage du modeleur, celui-ci peut être démarré une première fois sans le paramètre `-templategenerator`.

Note : Le lancement de la génération peut être fait depuis un ordonnanceur de tâches.

Note : Afin d'optimiser les temps de réponse, le temps de latence entre le serveur SmartEA et la machine qui démarre la génération doit être faible.

5.11.2. Paramètres

Différents paramètres sont utilisés par le modeleur servant à la génération basée sur des templates :

- `-templategenerator` : permet d'indiquer que le modeleur doit démarrer en mode générateur de diagrammes basé sur des templates.
- `-nosplash` : permet d'indiquer que le modeleur démarré pour la génération ne doit pas faire apparaitre le splashscreen.
- `-u [userid]` (paramètre requis) : permet d'indiquer l'identifiant de l'utilisateur (ou compte de service) qui execute la génération de diagrammes.
- `-p [mot de passe]` (paramètre requis) : permet d'indiquer le mot de passe de l'utilisateur (ou compte de service) qui execute la génération de diagrammes.
- `-project=[identifiant de projet]` (paramètre requis) : permet d'indiquer l'identifiant du projet sur lequel la génération est demandée. Il n'est possible de renseigner qu'un seul projet.

- `-branch=[identifiant de branche]` (paramètre requis) : permet d'indiquer l'identifiant de la branche sur laquelle la génération est demandée. Il n'est possible de renseigner qu'une seule branche.
- `-prism=[identifiant de prisme]` (paramètre requis) : permet d'indiquer l'identifiant du prisme avec lequel la génération est demandée.
- `-templateIds [identifiants de templates]` : permet d'indiquer une liste d'identifiants de templates pour lesquels exécuter la génération. Le mot clés réservé `ALL` permet d'indiquer que la génération doit être faite pour tous les template du projet.
- `-excludedTemplateIds [identifiants de templates]` : permet d'indiquer une liste d'identifiants de templates pour lesquels ne pas exécuter la génération.
- `-templateFolderIds [identifiants de répertoire de templates]` : permet d'indiquer une liste d'identifiants de répertoires de templates pour lesquels exécuter la génération.

Exemples :

Génération des diagrammes correspondant à une liste de templates donnés en paramètre pour un projet donné et une branche donnée :

```
Obeo-SmartEA-Modeler.exe -nosplash -templategenerator -u afontaine -p 123 -
project=voyagediscount -branch=Master -prism=EntArch -templateIds id1 id2
```

Génération des diagrammes correspondant à tous les templates d'un projet donné et d'une branche donnée :

```
Obeo-SmartEA-Modeler.exe -nosplash -templategenerator -u afontaine -p 123 -
project=voyagediscount -branch=Master -prism=EntArch -templateIds ALL
```

Génération des diagrammes correspondant aux templates des répertoires donnés en paramètre pour un projet donné et une branche donnée :

```
Obeo-SmartEA-Modeler.exe -nosplash -templategenerator -u afontaine -p 123 -
project=voyagediscount -branch=Master -prism=EntArch -templateFolderIds idFolder1 idFolder2
```

Génération des diagrammes correspondant aux templates des répertoires donnés en paramètre pour un projet donné et une branche donnée à l'exception de certains templates :

```
Obeo-SmartEA-Modeler.exe -nosplash -templategenerator -u afontaine -p 123 -
project=voyagediscount -branch=Master -prism=EntArch -templateFolderIds idFolder2 -
excludedTemplateIds id1 id2
```

Génération de tous les diagrammes pour un projet donné et une branche donnée à l'exception de certains templates :

```
Obeo-SmartEA-Modeler.exe -nosplash -templategenerator -u afontaine -p 123 -
project=voyagediscount -branch=Master -prism=EntArch -templateIds ALL -excludedTemplateIds id1
 id2
```

5.11.3. Codes de retour

Lorsque le modeleur termine son traitement un code de retour est envoyé :

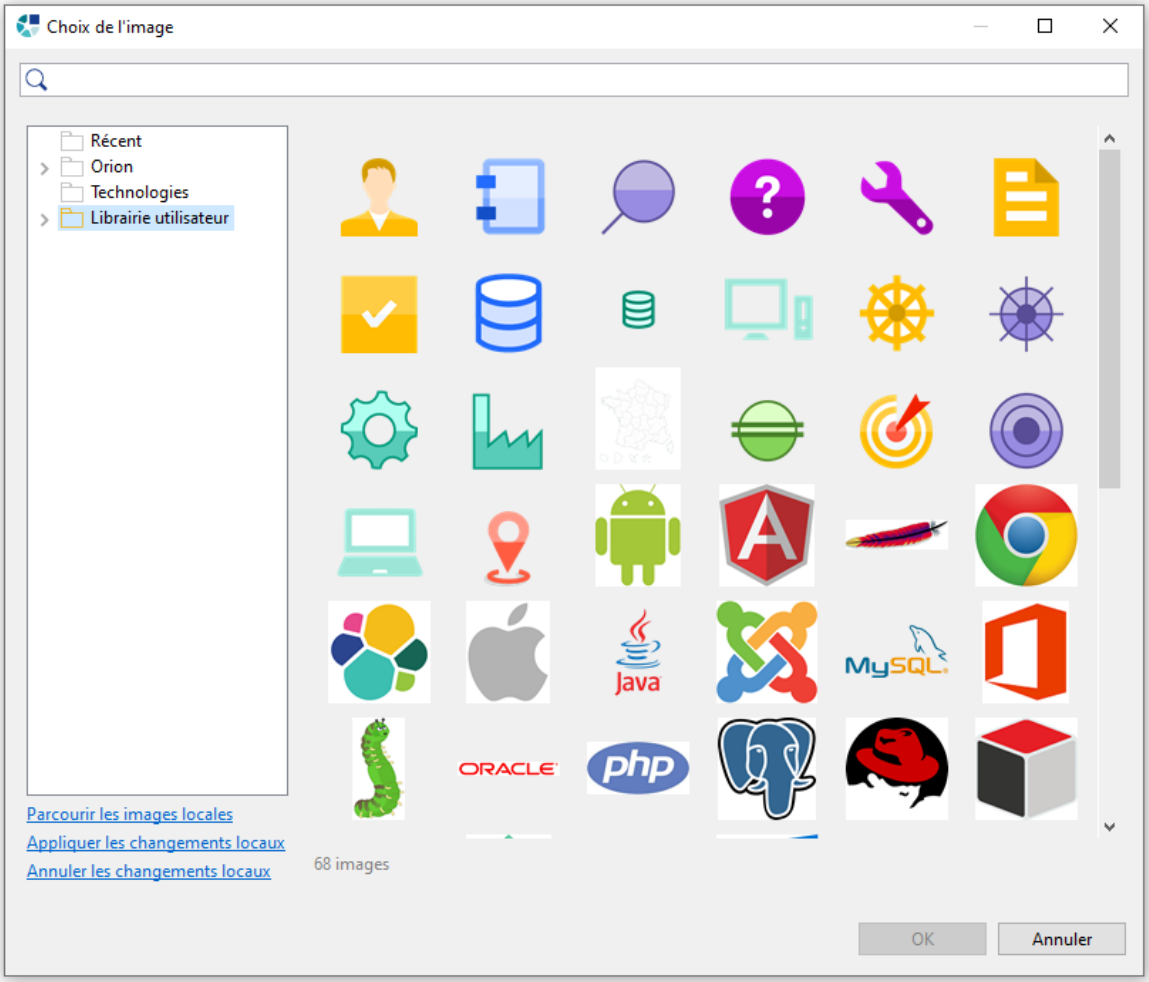
- 0 : génération réussie.
- 1 : exception inattendue.
- 11 : erreur de configuration.
- 12 : exception lors de la connexion.
- 13 : licence invalide.
- 14 : génération impossible car un éditeur est ouvert.
- 15 : erreur du moniteur de progression.
- 21 : erreur d'authentification.
- 22 : génération partielle. Certains objets sont verrouillés ou les autorisations ne permettent pas l'accès aux objets nécessaires au processus de génération. (autorisations).

Si le code de retour est différent de 0, le fichier `.log` se trouvant dans le répertoire `workspace/.metadata` vous donnera des détails sur le problème rencontré lors de la génération des diagrammes.

5.12. Librairie d'images

Dans les diagrammes SmartEA il est possible d'utiliser des images spécifiques. Le dialogue de sélection d'image permet d'importer une image, mais il est également possible de déposer une ou plusieurs images directement sur le serveur, dans le répertoire `data\images`, qui seront par la suite accessible via le dialogue de sélection.Au démarrage, le serveur synchronise les images accessibles avec les images du répertoire `data\images`.Si vous souhaitez synchroniser ce répertoire en cours d'utilisation, utilisez l'action **Reconstruire l'index** de la page d'administration transverse, puis relancez les modeleurs afin qu'ils se synchronisent avec le serveur.

Il est également possible de modifier la librairie d'images depuis un modeleur. Pour cela ajoutez la feature *Images Management*.



Le dialogue de sélection d'image fournit alors trois nouvelles actions :

- **Parcourir les images locales** : ce lien ouvre le répertoire local au modeleur ou sont stockées les images récupérées depuis le serveur. Vous pouvez ajouter, supprimer des images, réorganiser les répertoires ici. Les changements faits ne sont pas directement appliqués au serveur. Les images supprimées ne seront plus disponible dans le modeleur courant jusqu'à la prochaine synchronisation.
- **Appliquer les changements locaux** : cette action synchronise le répertoire local, où d'éventuels changements ont été faits, vers le serveur. Les changements réalisés seront communs à tous les utilisateurs de SmartEA qui devront relancer leurs modeleurs pour en bénéficier.

- **Annuler les changements locaux** : cette action synchronise le répertoire local depuis le serveur, annulant donc tout changement non synchronisé.

5.13. Personnalisation de la page d'authentification

Des options dans le fichier «etc/application.conf» permettent de modifier l'affichage de la page d'authentification :

- changer l'image d'entête : `login.logo`
- supprimer l'icône du navigateur : `login.smartea.icon`
- ajouter un avertissement : `login.disclaimer`.

Pour plus de détails, reportez vous à la description de ces options dans la section etc/application.conf.

5.14. Personnalisation de la recherche rapide

Par défaut, les recherches dans la barre de menu principale suivent la procédure suivante :

1. recherche de l'expression stricte
2. si il n'y a pas de résultats, ajout d'un wildcard en suffixe pour rechercher tous les éléments dont le nom commence par l'expression
3. si il n'y a pas de résultats, ajout d'un wildcard en préfixe pour rechercher tous les éléments dont le nom commence ou finissent par l'expression
4. si il n'y a pas de résultats, recherche dans le noms de types
5. si il n'y a pas de résultats, recherche dans les propriétés dynamiques

Une fois la recherche effectuée, sur la page de recherche, un utilisateur peut modifier l'expression résultant de cette procédure en ajoutant les wildcards nécessaires à la recherche.

Il est possible de faire en sorte que la recherche rapide positionne par défaut des wildcards aidant à la recherche.Pour ceci vous pouvez modifer l'option `quick.search.wildcards` du fichier etc/application.conf. qui supporte les modes suivants :

- **AUTO** (mode par défaut) : reste sur le mode automatique décrit précédemment
- **LEFT** : ajoute systématiquement un wildcard à la gauche des mots de l'expression
- **RIGHT** : ajoute systématiquement un wildcard à la droite des mots de l'expression
- **BOTH** : ajoute systématiquement un wildcard à la gauche et à la droite des mots de l'expression

Par exemple, si un référentiel contient les objets suivants :

```
* Test
* MyTest
* Test1
```

La recherche rapide de la chaine de caractères «Test» donnera les résultats suivants selon le mode choisi :

```
* *AUTO* : Test
* *LEFT* : MyTest, Test
* *RIGHT* : Test, Test1
* *BOTH* : MyTest, Test, Test1
```

5.15. Personnalisation des tables de résultats des Smart Requests

Afin de personnaliser les tables de résultats des Smart Requests les paramètres suivants sont disponibles dans le fichier «etc/application.conf» :

- `tables.page.lengthMenu` : Définit les longueurs de page pour la pagination des résultats des Smart Requests (valeurs séparées par une virgule et le tout entre crochets []). Une longueur de -1 correspond à une option d'affichage de tous les résultats. Par exemple [10, 100, -1] indique que les utilisateurs peuvent choisir entre une longueur de page de 10, de 100 ou de ne pas paginer les résultats.
- `tables.page.defaultLength` : Définit la longueur de page par défaut pour les résultats de Smart Requests.

- `tables.filter.threshold` : Par une valeur entre 0 et 1 vous pouvez définir le seuil d'affichage des filtres sur les colonnes des résultats de Smart Requests en fonction de la proportion de valeurs uniques qu'elles contiennent.
  - Un seuil de 0.5 signifie que les filtres s'afficheront si au moins 50% des valeurs de la colonne sont uniques,
  - Un seuil de 1 signifie que les filtres s'afficheront systématiquement,
  - Un seuil de 0 signifie que les filtres ne s'afficheront jamais.

Pour plus de précisions vous pouvez consulter cette page

5.16. Restriction de l'accès aux sites consultables depuis le Modeleur

L'option `rcp.hosts.authorized` dans le fichier «etc/application.conf» permet de limiter l'accès aux sites consultables depuis le modeleur.

Ce paramètre permet de définir une liste blanche des sites (host) que les modeleurs ont le droit de consulter.

Si un utilisateur tente d'accéder à un site n'appartenant pas à cette liste à partir d'un modeleur, alors un message d'erreur indiquera à l'utilisateur que l'accès n'est pas autorisé.

Si la liste est vide ou si ce paramètre n'est pas défini alors il n'y a aucune limitation d'accès.

Dans l'exemple suivant, les utilisateurs des modeleurs ont uniquement accès aux sites web de Obeo, de Google et bien sûr au serveur SmartEA (implicite).

```
rcp.hosts.authorized=www.obeo.fr, www.obeosoft.com, www.google.fr, ogs.google.fr
```

5.17. Ajout d'un texte en pied de page des pages web

Il est possible d'ajouter un texte en pied de page des pages web. Ce texte peut être du code html.

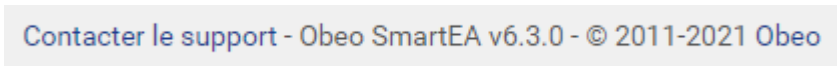
Si vous souhaitez activer cette fonctionnalité, vous dever modifier le fichier `etc/application.conf` en renseignant le paramètre `footer.specific`.

Exemple :

```
footer.specific=<a href="https://news.obeosoft.com/" target="_blank">Actualit&eacute;s Obeo</
a>
```

5.18. Ajout d'un lien en pied de page des pages web permettant d'envoyer un email

Il est possible d'ajouter un lien en pied de page des pages web qui permet d'ouvrir le lecteur d'emails de l'utilisateur avec un email pré rempli à envoyer.



Si vous souhaitez activer cette fonctionnalité, vous dever modifier le fichier `etc/application.conf` en renseignant les éléments suivants :

- `support.contact` : positionnez la valeur à `true` pour activer la fonctionnalité.
- `support.contact.email` : permet de renseigner l'adresse email du destinataire.
- `support.contact.subject` : permet de renseigner le sujet par défaut de l'email.
- `support.contact.body` : permet de renseigner le contenu par défaut de l'email.

Les variables suivantes peuvent être utilisées pour `support.contact.subject` et `support.contact.body` :

- `$location` : l'url de la page courante.

- `$version` : la version de SmartEA.
- `$user` : l'identifiant de l'utilisateur courant.

La variable suivante peut être utilisée pour `support.contact.body`

- `$cr` : retour chariot.

Exemple :

```
support.contact=true
support.contact.label=Contacter le support
support.contact.email=foo@obeo.fr
support.contact.subject=Référentiel d'Entreprise - Remarque de $user
support.contact.body=Bonjour, $cr$cr <Ajoutez vos remarques ici> $cr$cr$cr Version : SmartEA
 $version $cr Utilisateur : $user $cr Contexte : $location $cr$cr  $cr$cr Cordialement, $cr
```

5.19. Sauvegarde du serveur

Obeo recommande soit une sauvegarde régulière de la base de données et du répertoire de l'application serveur, soit une sauvegarde complète de la machine virtuelle du serveur SmartEA.

La stratégie de rétention des sauvegardes peut être définie comme suit :

- Les sauvegardes journalières sont conservées deux semaines,
- Une sauvegarde par semaine est conservée pendant 1 mois,
- Une sauvegarde mensuelle est conservée pendant 6 mois.

Ces mesures de sauvegarde technique peuvent être complétées au besoin par un export régulier des projets (action scriptable) permettant une restauration autonome par l'administrateur fonctionnel (voir Section Administration en ligne de commande).

En fonction de la criticité des données hébergées et de leur usage, cette fréquence de sauvegarde et la durée de rétention doivent être en accord avec la politique de sécurité de l'entreprise pour assurer un équilibre raisonnable d'atténuation du risque par rapport au coût.

5.19.1. Exemple de script de sauvegarde de la base de données PostgreSQL

Ci-dessous un exemple de script en Python permettant de sauvegarder la base de données PostgreSQL.

```
#!/usr/bin/python3
import time,os,glob
databaseName='smarteadb740'
backupFolder='/Shares/Backups'
print("Start backup of the database "+databaseName+". Date: "+time.asctime
 (time.localtime(time.time())) )
str_date=time.strftime("%Y_%m_%d",time.localtime(time.time()))
backupName='backup_'+databaseName+'__'+str_date+'.dump.gz'
dumpCommand='su - postgres -c "pg_dump ' +databaseName+ '"'
# dump
test=os.system(dumpCommand+"| gzip > /tmp/"+backupName )
print(time.asctime (time.localtime(time.time())) )
if test==0:
    os.system("cp /tmp/"+backupName+" "+backupFolder)
    print('Backup of the database '+databaseName+' done.')
    print('In '+backupFolder+'/'+backupName)
else:
    print('Error. Unable to backup the database '+databaseName+'.')
    sys.exit()
os.system("rm /tmp/"+backupName)
# Remove old backups
backupsToRetain=30
backupsToRetain=-backupsToRetain
ldel=glob.glob(backupFolder+'/backup_'+databaseName+'*')
```

```
ldel.sort()
print("Delete old backups")
for fic in ldel[:backupsToRetain]:
    os.system("/bin/rm -f "+fic)
print("Backup end")
```

Vous pouvez ensuite déclencher ce script de manière périodique grâce à un ordonnanceur.

Exemple avec l'outil `cron` sous Linux (déclenchement tous les jours à 20h) :

```
# Backup DB
00 20   * * *   root /usr/local/bin/SmarteaDBBackup.py
```

5.20. Journaux

Le journal d'un modeleur SmartEA se trouve dans le répertoire `workspace/.metadata` dans le fichier `.log`.

Les journaux du serveur SmartEA se trouvent dans le répertoire `data/logs`. Ils sont au nombre de 5.

- smartea.log : ce fichier enregistre les informations de démarrage du serveur, les services appellés, les erreurs, les avertissements, etc.
- user-accesses.log : ce fichier enregistre les accès des utilisateurs aux pages web du serveur.
- write-operations.log : ce fichier enregistre les opérations d'écriture effectuées sur les données sémantiques.
- functional-conf-modifications.log : ce fichier enregistre les modifications de configuration fonctionnelle.
- system-informations.log : ce fichier enregistre des informations sur l'état et les performances du serveur.

Chacun de ces fichiers fournit des informations pour surveiller et comprendre le fonctionnement du serveur, ainsi que pour identifier les problèmes potentiels et les menaces à la sécurité.

Les journaux `user-accesses.log`, `write-operations.log`, `functional-config-modifications.log` et `system-informations.log` ont chacun un format spécifique identique pour chaque ligne. Il est ainsi possible de facilemment les exploiter dans un outil tierce.

Chacun de ces 4 journaux peut être activé ou désactivé selon les besoins. Les paramètres d'activation/désactivation se trouvent dans le fichier `etc/application.conf`

- analytics.userAccesses=true|false : permet d'activer ou désactiver le journal qui enregistre les accès des utilisateurs aux pages web du serveur.
- analytics.writeOperations=true|false : permet d'activer/désactiver le journal qui enregistre les opérations d'écriture effectuées sur les données sémantiques.
- analytics.functionalConfModifications=true|false : permet d'activer/désactiver le journal qui enregistre les opérations d'écriture effectuées sur les données sémantiques.
- analytics.systemInfos=true|false : permet d'activer/désactiver le journal qui enregistre les informations sur l'état et les performances du serveur.

Par défaut ces 4 journaux sont désactivés.

4 préférences permettent de préciser les séparateurs à utiliser entre les éléments d'une ligne de chacun de ces 4 journaux.

- analytics.separator : permet de définir le séparateur à utiliser entre chaque élément d'une ligne. Par défaut le caractère `,` est utilisé.
- analytics.separator.list.start : permet de préciser comment est indiqué le début d'une liste pour un élément de type liste. Par défaut le caractère `[` est utilisé.
- analytics.separator.list.end : permet de préciser comment est indiqué le début d'une liste pour un élément de type liste. Par défaut le caractère `]` est utilisé.
- analytics.separator.list.separator : permet de définir le séparateur à utiliser entre chaque élément de type liste. Par défaut le caractère `|` est utilisé.

Exemple : en utilisant la configuration par défaut, une ligne d'un de ces 4 journaux à la forme suivante :

```
2024.02.08.18.04.56.300+0100,voyagediscount,Master,afontaine,value#1,value#2,[value#3.1|
value#3.2|value#3.3],value#4
```

La préférence `analytics.systemInfos.frequency` du fichier `application.conf` permet de préciser en minutes la fréquence à laquelle l'état du serveur doit être enregistré dans le journal system-informations.log. Il est conseillé de ne pas fixer une valeur trop petite pour ne pas surcharger inutilement le serveur.La valeur par défaut est de 15 minutes (15). La valeur minimale est 1 minute.

La préférence `analytics.hideValues` permet d'indiquer si la valeur des éléments sémantiques doit ou non apparaitre dans les journaux. Par défaut la valeur est `false`.

La préférence `analytics.userAgent` permet d'indiquer si dans le journal des accès faits par les utilisateurs, le User Agent du navigateur doit ou non apparaitre. Par défaut la valeur est `false`.

Le nom des journaux `user-accesses.log`, `write-operations.log`, `functional-config-modifications.log` et `system-informations.log` peut être modifié dans le fichier `etc/application_log.conf`.

Ce fichier permet également de modifier la taille maximum du fichier de log, le nombre de sauvegardes à conserver avant rotation du fichier ainsi que le format de l'horodatage.

Pour chacun des 4 journaux vous trouverez ainsi dans le fichier `etc/application_log.conf` 4 lignes ayant le pattern suivant :

- log4j.appender.smarteaAnalytics*Appender.layout.ConversionPattern : permet de préciser le format de l'horodatage.
- log4j.appender.smarteaAnalytics*Appender.File : permet de préciser le nom du fichier de log (`${smartea.data.directory}` est le répertoire `data/` du serveur).
- log4j.appender.smarteaAnalytics*Appender.MaxFileSize : permet de préciser la taille maximum du fichier de log.
- log4j.appender.smarteaAnalytics*Appender.MaxBackupIndex : permet de préciser le nombre de fichiers de log à conserver lors de la rotation.

5.20.1. user-accesses.log

Le format du fichier `user-accesses.log` est le suivant :

- Colonne 1 : Horodatage de l'évènement
- Colonne 2 : Type de l'évènement
  - Home_Page : accès à la page d'accueil.
  - Semantic_Object_Page : accès à la page de détails d'un objet sémantique.
  - Page_Tab_Change : accès à un onglet d'une page de détails d'un objet sémantique.
  - Folder_Page : accès à la page de détails d'un répertoire.
  - Artifact_Page : accès à la page d'un artefact.
  - Artifact_Create : création d'un artefact.
  - Artifact_Delete : suppression d'un artefact.
  - Artifact_Download : téléchargement d'un artefact.
  - Artifact_Duplicate : duplication d'un artefact.
  - Artifact_Rename : renommage d'un artefact.
  - Artifact_Unpublish : suppression de la publication d'un artefact.
  - Login : connexion.
  - Logout: deconnexion.
  - Branch_Create : création d'une branche.
  - Branch_Restore : restauration d'une branche.
  - Branch_Delete : suppression d'une branche.
  - Compare_Create : comparaison de deux branches.
  - Search : recherche textuelle.

    - Search_Advanced : recherche avancée.
- Colonne 3 : user agent si celui-ci est enregistré (paramètre `analytics.hideValues`), sinon X.
- Colonne 4 : identifiant de la session de l'utilisateur.
- Colonne 5 : identifiant de l'utilisateur.
- Colonne 6 : identifiant du projet.
- Colonne 7 : identifiant de la branche.
- Colonne 8 : identifiant du prisme.
- Colonnes 9 à 13 : selon le type de l'évènement.
- Colonne 14 : temps d'éxécution backend du service.

Pour les évènements Login et Logout sont enregistrés uniquement les colonnes 1, 2, 4, 5 et 14.

Pour l'évènement Home_Page rien n'est enregistré dans les colonnes 9 à 13 (colonnes réservées).

Pour les évènements Semantic_Object_Page, Folder_Page, Artifact_Page, Page_Tab_Change, Artifact_Create, Artifact_Delete, Artifact_Download, Artifact_Duplicate, Artifact_Rename et Artifact_Unpublish est enregistré en :
- Colonne 9 : l'identifiant physique de l'objet.
- Colonne 10 : l'identifiant logique de l'objet.
- Colonne 11 : le type de l'objet et ses identifiants de stéréotypes le cas échéant.
- Colonne 12 : le label de l'objet selon la valeur du paramètre `analytics.hideValues` (sinon X).
- Colonne 13 : colonne réservée sans valeur pour tous les événements à l'exception de l'événement Page_Tab_Change où l'index de l'onglet cliqué est enregistré.

Pour les évènements Branch_Create, Branch_Restore, Branch_Delete est enregistré :
- Colonne 9 : l'identifiant de la branche.
- Colonne 10 : colonne réservée sans valeur.
- Colonne 11 : colonne réservée sans valeur.
- Colonne 12 : colonne réservée sans valeur.
- Colonne 13 : colonne réservée sans valeur.

Pour l'évènement Compare_Create est enregistré :
- Colonne 9 : l'identifiant de la branche origine.
- Colonne 10 : l'identifiant de la branche cible.
- Colonne 11 : colonne réservée sans valeur.
- Colonne 12 : colonne réservée sans valeur.
- Colonne 13 : colonne réservée sans valeur.

Pour les évènements Search et Search_Advanced est enregistré :
- Colonne 9 : la requête selon la valeur du paramètre `analytics.hideValues` (sinon X).
- Colonne 10 : colonne réservée sans valeur.
- Colonne 11 : colonne réservée sans valeur.
- Colonne 12 : colonne réservée sans valeur.
- Colonne 13 : colonne réservée sans valeur.

5.20.2. write-operations.log

Le format du fichier `write-operations.log` est le suivant :
- Colonne 1 : Horodatage de l'évènement
- Colonne 2 : identifiant du projet.
- Colonne 3 : identifiant de la branche.
- Colonne 4 : identifiant de l'utilisateur.
- Colonne 5 : la liste des ajouts.

- Colonne 6 : la liste des modifications.
- Colonne 7 : la liste des suppressions.

5.20.3. functional-conf-modifications.log

Le format du fichier `functional-config-modifications.log` est le suivant :

- Colonne 1 : Horodatage de l'évènement
- Colonne 2 : identifiant du projet.
- Colonne 3 : identifiant de la branche.
- Colonne 4 : identifiant de l'utilisateur.
- Colonne 5 : la liste des ajouts.
- Colonne 6 : la liste des modifications.
- Colonne 7 : la liste des suppressions.

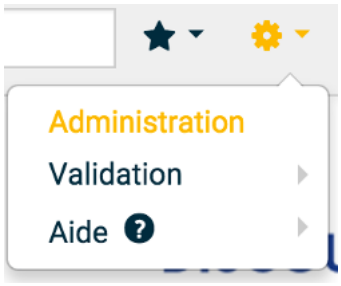
5.20.4. system-informations.log

Le format du fichier `system-informations.log` est le suivant :

- Colonne 1 : Horodatage de l'évènement.
- Colonne 2 : le type de l'évènement :
  - Start : enregistrement au démarrage du serveur.
  - Tick : enregistrement en cours d'exécution selon la fréquence précisée par `analytics.systemInfos.frequency`.
  - Stop : enregistrement à l'arrêt du serveur.
- Colonne 3 : la charge moyenne de la machine sur la dernière minute (non accessible pour les machines Windows, valeur -1 dans ce cas).
- Colonne 4 : l'espace disque disponible.
- Colonne 5 : les identifiants des sessions RCP
  - Note : une connexion RCP d'un utilisateur se traduit pas deux sessions : une sur la branche interne et une autre sur la branche sémantique. 2 sessions apparaissent dans le log mais un seul jeton est pris dans ce cas.
- Colonne 6 : les identifiants des sessions Web en mode contributeur.
- Colonne 7 : la mémoire heap.
- Colonne 8 : la mémoire non heap.

6. Administration SmartEA

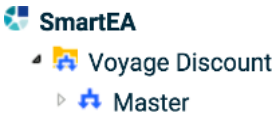
Les utilisateurs possédant le droit d'administration ont accès au menu *Administration* de la barre d'outils.



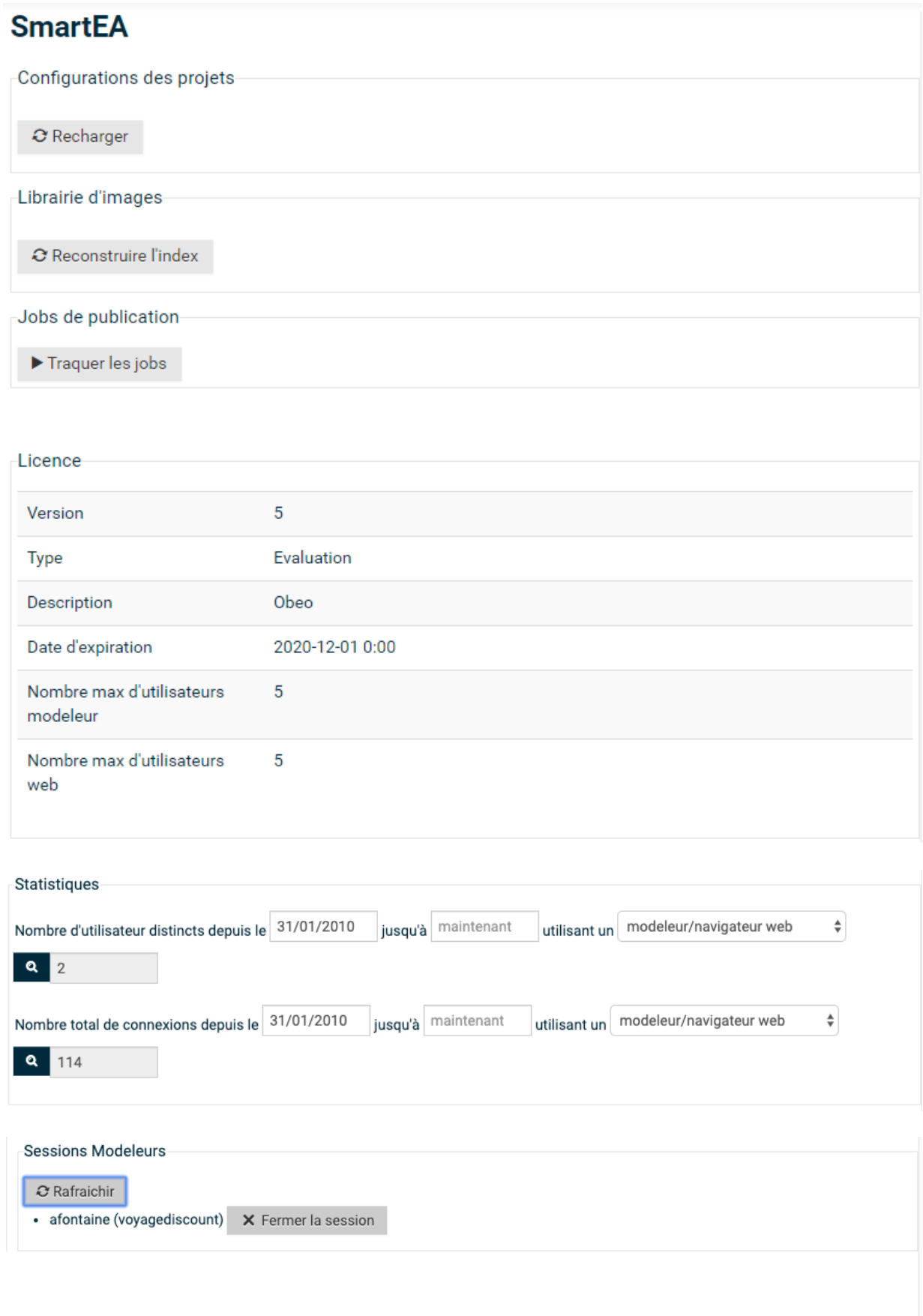
Note **:** L'accès au menu d'administration nécessite d'être connecté avec un utilisateur ayant les droits d'administration.

6.1. Administration transverse

Le menu «SmartEA» va afficher l'ensemble des fonctionnalités d'administration transverses.



Fonctionnalités d'administration transverses:

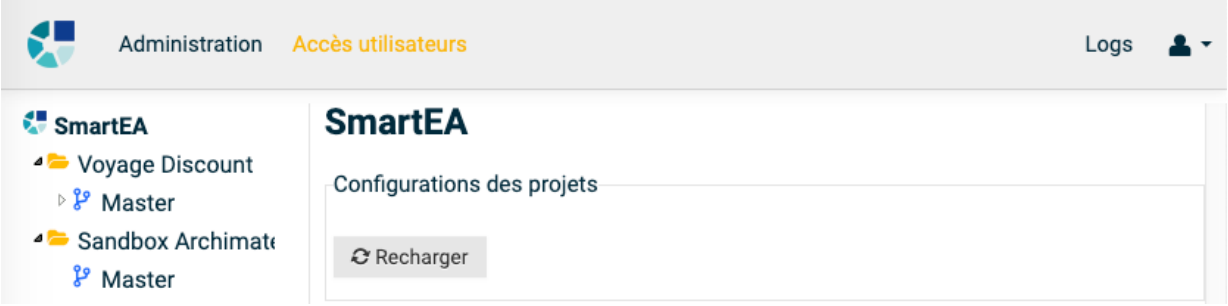


- Configurations des projets : Permet de recharger les fichiers de configuration des projets.

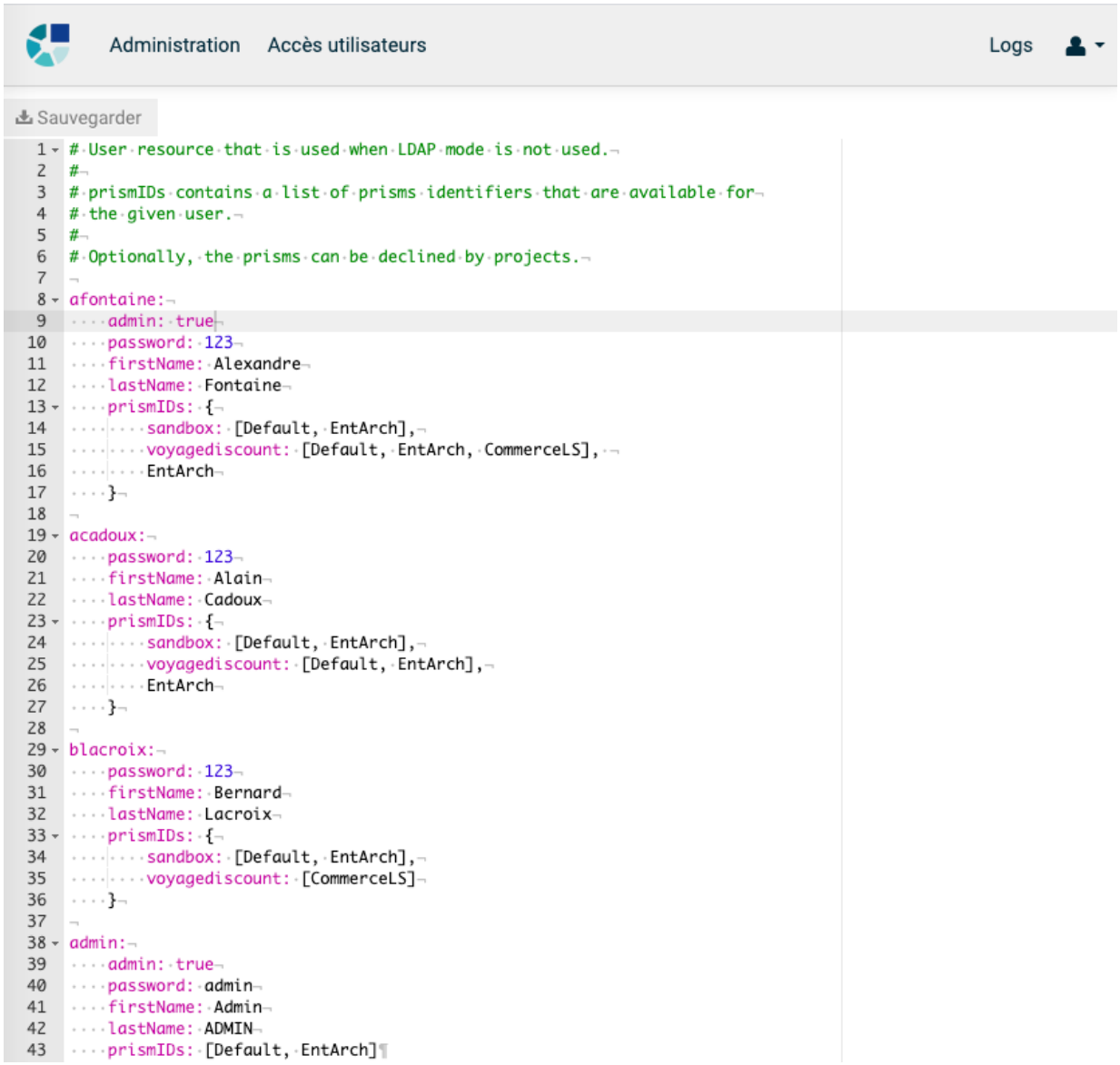
- Librairie d'images : Permet de reconstruire l'index des images déposées sur le serveur
- Jobs de publication : Permet de suivre le statut des demandes de publication (programmées/en cours/ terminées).
- licence : Affiche le statut de la licence.
- Statistiques : Permet d'afficher le nombre distincts d'utilisateur ou de connexion à l'outil.
- Sessions Modeleurs : Permet de lister les sessions modeleurs et de les fermer à distance. Le fait de fermer une session modeleur peut provoquer la perte des données non sauvegardées.

6.2. Accès utilisateurs

Pour éditer les accès utilisateurs, utilisez le menu de la page d'administration :



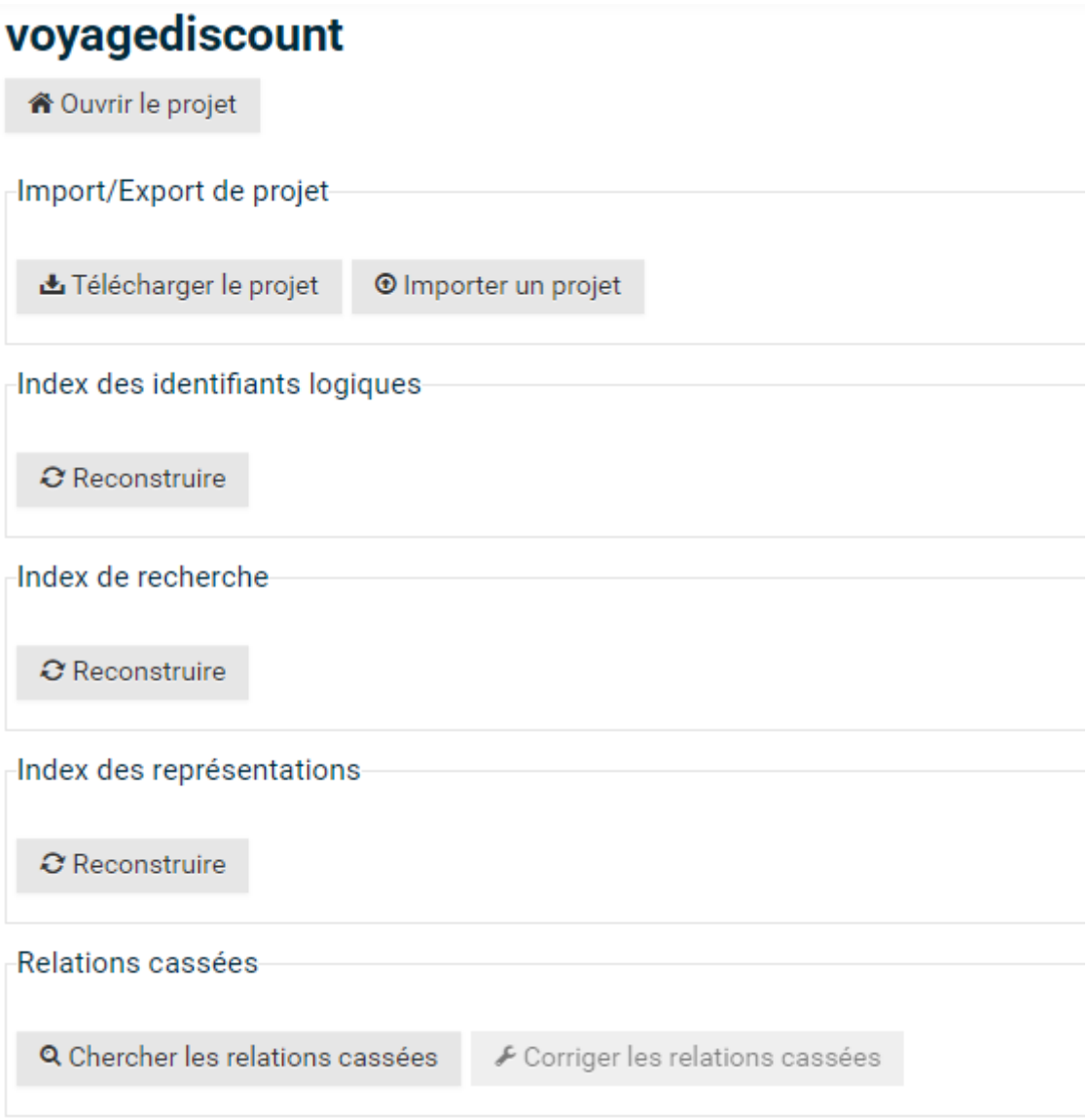
Un éditeur est alors disponible.Suivant la configuration des accès utilisateurs, la structure à respecter correspond à la section user.yml ou profiles.yml.



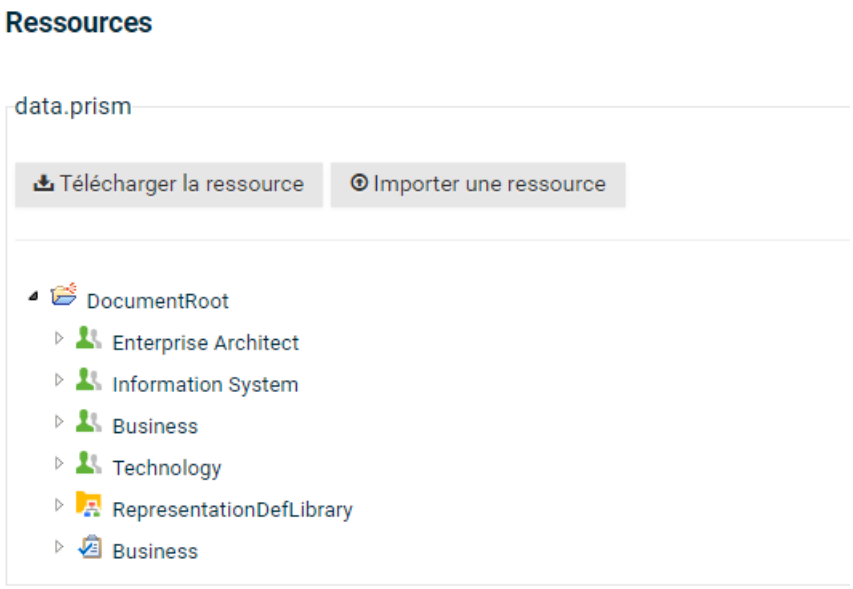
6.3. Administration des projets

Le menu «Voyage Discount» va afficher l'ensemble des fonctionnalités d'administration du projet.

Fonctionnalités d'administration d'un projet :

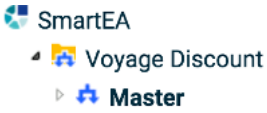


- Import/Export d'un projet : Permet de télécharger ou d'écraser le projet courant. (Le résultat est un fichier zip). ***Note*:** Ce téléchargement peut être utilisé afin de sauvegarder l'état du projet. L'écrasement peut être utilisé afin de restaurer l'état d'un projet._
- Index des identifiants logiques : Permet de régénérer l'index sur lequel se base *Obeo SmartEA* pour identifier les objets sémantiques et les artefacts.
- Index de recherche : Permet de régénérer l'index sur lequel se base le moteur de recherche.
- Index des représentations : Permet de régénérer l'index sur lequel se base *Obeo SmartEA* pour indiquer quelle représentation contient quel objet (section *Related Representations* sur la page de détails d'un objet).
- Relations cassées : Permet de chercher et de nettoyer les références cassées entre ressources.
- Prismes : Permet de télécharger ou d'écraser la ressource des prismes.



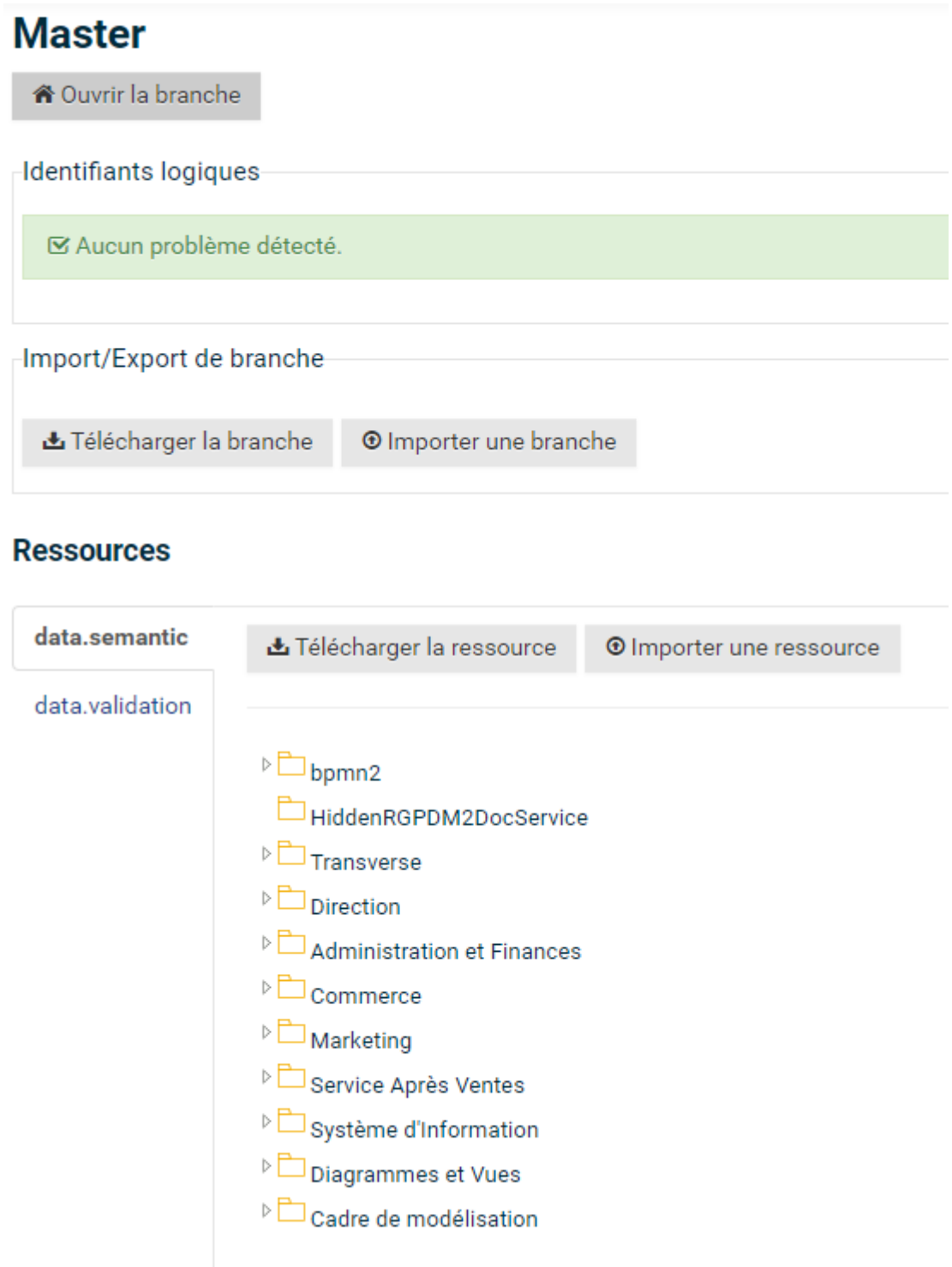
6.4. Administration des branches

Un projet dispose toujours d'une branche **Master** :



Il peut avoir des sous-branches :

Si vous sélectionnez une branche, des outils dédiés sont disponibles :



- Identifiants logiques : Permet de détecter automatiquement les identifiants logiques dupliqués ou invalides. **Les identifiants logiques non indexés ne seront pas détectés.**
- Import/export de branche : Permet de télécharger (le résultat est un fichier zip) ou d'écraser la branche sélectionnée.
- Ressources : Vous pouvez télécharger ou écraser toutes les ressources disponibles dans la branche courante. Si vous écrasez une ressource, vous risquez de casser des références entre ressources. (L'outil **Relations cassées** vous permettra de les détecter et de les nettoyer)

6.5. Administration en ligne de commande

Il est possible de lancer certaines commandes d'administration via un script, ce qui peut être utile pour automatiser certaines tâches. Il faut pour cela utiliser l'outil curl.

- Export d'un projet :

```
curl -u <login>:<password> <http|https>://<host>:<port>/smartea/<project_id>/service/admin/
resources/project/download > <filename>.zip
```

- Import d'un projet :

```
curl -u <login>:<password> <http|https>://<host>:<port>/smartea/<project_id>/service/admin/
resources/project/upload -F 'resourceFile=@<filename>.zip'
```

7. FAQ

7.1. FAQ Serveur

7.1.1. Pourquoi le serveur ne démarre-t-il pas ?

Consultez le dossier `configuration`, pour accéder au fichier de traces du démarrage en échec.

Si le problème arrive suite à l'installation d'un module, il est possible que le module soit mal installé et que les traces évoquent la cause. A l'installation du module, avez-vous bien crée un dossier parent ?

7.1.2. Que faire si un utilisateur n'a accès à aucun(e) prisme / branche / projet ?

Dans le cas ou vous utilisez l'authentification par défaut (user.yml) vérifiez la configuration et notamment que l'utilisateur en question à au moins un(e) prisme / branche / projet accessible.

Dans le cas ou vous utilisez l'authentification par LDAP, vérifiez la configuration et notamment que l'utilisateur en question à au moins un(e) prisme / branche / projet accessible.

7.1.3. Je n'arrive pas à accéder à la page d'administration !

Dans le cas ou vous utilisez l'authentification par défaut (user.yml) vérifiez que l'utilisateur est bien administrateur

Dans le cas ou vous utilisez l'authentification par LDAP vérifiez que l'utilisateur est bien administrateur

7.1.4. Je n'arrive pas à accéder à l'administration des prismes !

Il est possible que la fonctionnalité d'administration des prismes ne soit pas active. Référez-vous à la documentation

7.1.5. Que faire si ma licence est invalide ?

Reportez vous à la section sur les licences.

7.1.6. J'ai une erreur "Ident authentication failed for user «smarteainternal» !

Sur le serveur PostgreSQL 9.1, Vous devez vérifier que le fichier de configuration ph_hba.conf autorise bien la méthode de connexion md5 (http://docs.postgresqlfr.org/9.1/client-authentication.html)Le message d'erreur suivant correspond à ce problème de configuration :

```
Caused by: org.postgresql.util.PSQLException: FATAL: Ident authentication failed for user
 "smarteainternal"
        at
 org.postgresql.core.v3.ConnectionFactoryImpl.doAuthentication(ConnectionFactoryImpl.java:398)
        at
 org.postgresql.core.v3.ConnectionFactoryImpl.openConnectionImpl(ConnectionFactoryImpl.java:173)
        at org.postgresql.core.ConnectionFactory.openConnection(ConnectionFactory.java:64)
        at
 org.postgresql.jdbc2.AbstractJdbc2Connection.<init>(AbstractJdbc2Connection.java:136)
        at
 org.postgresql.jdbc3.AbstractJdbc3Connection.<init>(AbstractJdbc3Connection.java:29)
        at
 org.postgresql.jdbc3g.AbstractJdbc3gConnection.<init>(AbstractJdbc3gConnection.java:21)
        at
 org.postgresql.jdbc4.AbstractJdbc4Connection.<init>(AbstractJdbc4Connection.java:31)
        at org.postgresql.jdbc4.Jdbc4Connection.<init>(Jdbc4Connection.java:24)
        at org.postgresql.Driver.makeConnection(Driver.java:393)
        at org.postgresql.Driver.connect(Driver.java:267)
```

```
        at java.sql.DriverManager.getConnection(Unknown Source)
        at java.sql.DriverManager.getConnection(Unknown Source)
        at play.db.DBPlugin.onApplicationStart(DBPlugin.java:87)
        ... 4 more
```

7.1.7. J’ai une erreur «Connections could not be aquired from the underlying database» !

Si la base de données n’est pas sur la même machine, vous pouvez tenter un test ping sur l’IP afin de vérifier que la machine du serveur SmartEA à bien accès à la base de données.

Il est très probable que la configuration PosgreSQL n’autorise pas la connexion par mot de passe par défaut. La configuration du fichier pd_hba.conf est dans ce cas à consulter.

7.1.8. J’ai une erreur «An attempt by a client to checkout a Connection has timed out» !

Il y a sûrement une erreur dans un ou plusieurs fichiers de configuration au niveau des paramètres `db.xxx`.

7.1.9. Rien à faire, je n’arrive pas à me connecter avec PosgreSQL !

Si vous ne trouvez pas la source de l’erreur, vous pouvez consulter les traces de la base de données PosgreSQL.

7.1.10. Certaines ressources comme les images provenants de jar ne sont pas trouvés dans les UI web!

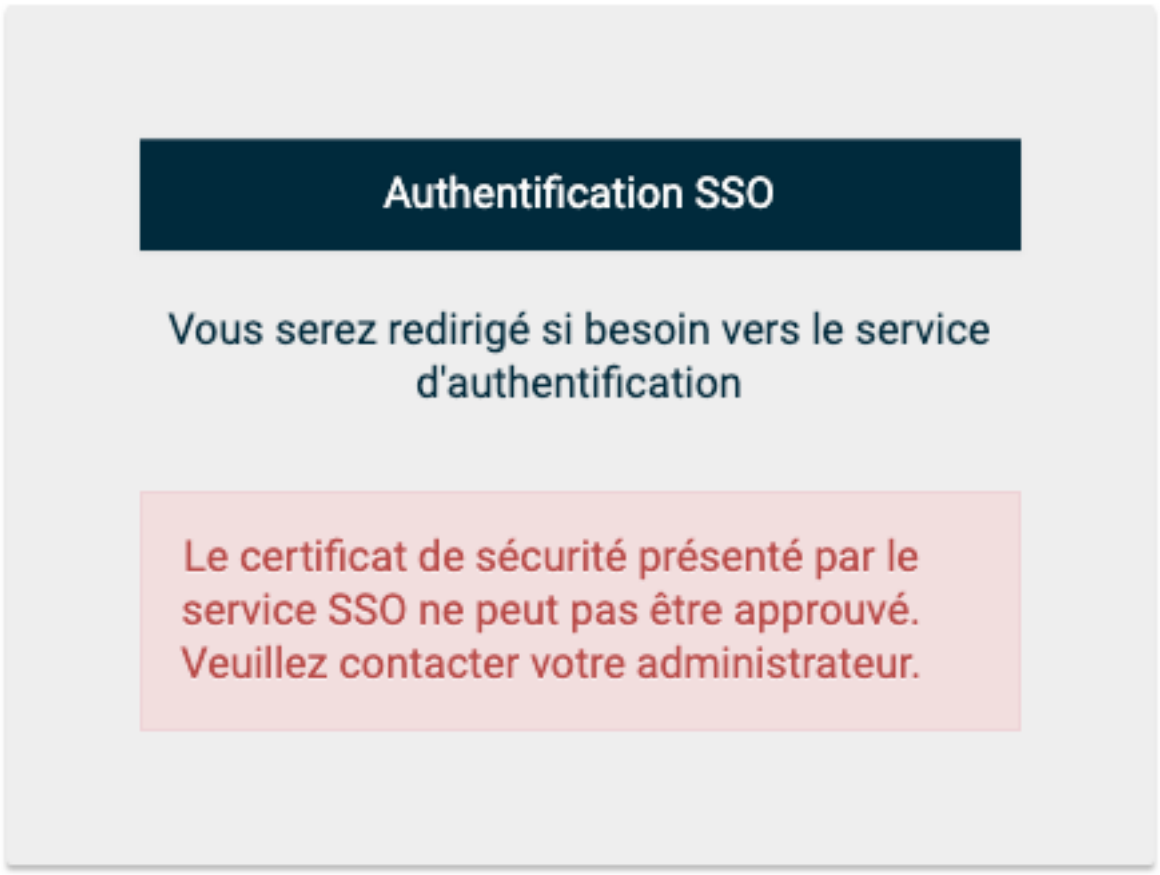
exemple : Erreur 404 pour static/images/icon_ObeoSmartEA_32.png

Vérifiez les droits et l’utilisateur dans le dossier (et sous-dossiers) d’installation du serveur. L’utilisateur qui lance le serveur doit avoir accès en lecture / écriture / exécution à l’intégralité du serveur

7.1.11. En mode SSO, je n’arrive pas accéder à SmartEA

J’ai un message du type :

```
Le certificat de sécurité présenté par le service SSO ne peut pas être approuvé. Veuillez
 contacter votre administrateur.
```



La JRE utilisée par le serveur SmartEA ne peut pas vérifier le certificat du site OpenID que vous avez configuré.Une opération de déclaration doit être faite à l'aide de la commande `keytool -import`.

Récupérez le certificat du site OpenID Connect, puis dans le dossier de la jre :

```
./bin/keytool -import -alias smartea -trustcacerts -keystore ./lib/security/cacerts -file ./
certif.pem -storepass changeit
```

7.1.12. Https : J'ai une erreur SSLHandshakeException au démarrage (Caused by: java.io.EOFException: SSL peer shut down incorrectly)

Si vous avez une trace de la forme suivante dans les logs cela peut être dû au fait que le niveau de l'algorithme de Peer signing digest de l'applicatif avec lequel SmartEA communique est trop faible.

```
Caused by: java.io.EOFException: SSL peer shut down incorrectly
        at java.base/sun.security.ssl.SSLSocketInputRecord.read(SSLSocketInputRecord.java:483)
        at java.base/
sun.security.ssl.SSLSocketInputRecord.readHeader(SSLSocketInputRecord.java:472)
```

```
        at java.base/
sun.security.ssl.SSLSocketInputRecord.decode(SSLSocketInputRecord.java:160)
        at java.base/sun.security.ssl.SSLTransport.decode(SSLTransport.java:111)
        at java.base/sun.security.ssl.SSLSocketImpl.decode(SSLSocketImpl.java:1506)
```

L'algorithme TLS 1.1 de Peer signing digest (pour l'intégrité des données échangées entre le client et le serveur lors d'une transaction TLS) est une concaténation de MD5 et SHA1.

A partir de TLS 1.2, l'algorithme est directement négocié entre le client et le serveur. La mise à jour de la RFC pour TLS 1.2 indique que le niveau de sécurité de cet algorithme devrait (SHOULD) être strictement supérieur à SHA1 (les produits comme OpenSSL 1.1.1 et OpenJDK 11, par exemple, implémente cette RFC).

Cependant certains applicatifs communiquant avec SmartEA (par exemple un annuaire) peuvent continuer d'utiliser uniquement du SHA1 comme algorithme de Peer signing digest en TLS 1.2, ce qui entraîne un handshake failure de la forme suivante :

```
javax.net.ssl.SSLHandshakeException: Remote host terminated the handshake
Caused by: java.io.EOFException: SSL peer shut down incorrectly
```

Un paramètre de configuration de la JVM, permet de rajouter l'algorithme SHA1 dans la liste de Peer signing digest. Cette approche permet d'éviter l'abaissement du niveau de protocole à TLS 1.1.

Exemple de paramétrage permettant de négocier une connexion TLS 1.2 (à rajouter dans les options de la JVM dans le fichier .ini du serveur et/ou du modeleur) :

```
-Djdk.tls.client.SignatureSchemes=rsa_pkcs1_sha1,rsa_pkcs1_sha256,rsa_pkcs1_sha384,
rsa_pkcs1_sha512,dsa_sha256,rsa_sha224,dsa_sha224,dsa_sha1,ecdsa_secp256r1_sha256,
ecdsa_secp384r1_sha384,ecdsa_secp521r1_sha512,ecdsa_sha224,ecdsa_sha1
```

7.1.13. Linux : Les fonctions d'export/import Excel génériques ne fonctionnent pas (polices de caractères non trouvées)

Les fonctions d'export et d'import génériques ne fonctionnent pas et génèrent la trace suivante dans le journal serveur :

```
java.lang.NullPointerException
at sun.awt.FcFontManager.getDefaultPlatformFont(FcFontManager.java:76)
at sun.font.SunFontManager$2.run(SunFontManager.java:443)
at java.security.AccessController.doPrivileged(Native Method)
at sun.font.SunFontManager.<init>(SunFontManager.java:386)
at sun.awt.FcFontManager.<init>(FcFontManager.java:35)
at sun.awt.X11FontManager.<init>(X11FontManager.java:57)
...
```

ou bien encore :

```
Caused by: java.lang.NullPointerException
    at java.desktop/sun.awt.FontConfiguration.getVersion(FontConfiguration.java:1288)
    at java.desktop/sun.awt.FontConfiguration.readFontConfigFile(FontConfiguration.java:225)
    at java.desktop/sun.awt.FontConfiguration.init(FontConfiguration.java:107)
    at java.desktop/sun.awt.X11FontManager.createFontConfiguration(X11FontManager.java:765)
    at java.desktop/sun.font.SunFontManager$2.run(SunFontManager.java:440)
    at java.base/java.security.AccessController.doPrivileged(Native Method)
    at java.desktop/sun.font.SunFontManager.<init>(SunFontManager.java:385)
    at java.desktop/sun.awt.FcFontManager.<init>(FcFontManager.java:35)
    at java.desktop/sun.awt.X11FontManager.<init>(X11FontManager.java:56)
    ... 36 more
```

Vérifiez si des polices de caractères sont installées sur votre serveur (la commande `fc-list` permet de lister les polices installées). Le JDK ne fournit aucune police de caractères pour Linux (contrairement à la version Windows).

Si il n'y a aucune police de caractères sur votre serveur, installez les polices suivantes : `Freetype`, `Fontconfig` et `DejaVu`.

```
Debian/Ubuntu/Mint: apt-get install libfreetype6 fontconfig fonts-dejavu
RHEL/CentOS/Fedora: yum install freetype fontconfig dejavu-sans-fonts
SLES/OpenSUSE: zypper install libfreetype6 fontconfig dejavu-fonts
```

Vous devez également copier/coller le fichier `fontconfig.properties` (répertoire `data/misc` du serveur) dans le répertoire `lib` de votre installation Java. Vérifiez que les chemins qui se trouvent dans ce fichier sont corrects (`fc-list` permet d'obtenir le chemin des polices installées).

Il est nécessaire de redémarrer le serveur SmartEA pour que la modification soit prise en compte.

7.1.14. La marguerite des relations d'une page de détail d'un objet met beaucoup de temps à s'afficher

Voir ici si vous utilisez une base de données PostgreSQL

7.2. FAQ Modeleur

7.2.1. Pourquoi le modeleur ne démarre-t-il pas ?

Essayez dans un premier temps avec un navigateur web. Avez vous accès à SmartEA ?Si oui il est possible que le serveur soit mal configuré.Si non il est possible que la configuration réseau entre le serveur et le modeleur soit à revoir.

7.2.2. Mon modeleur fonctionnait la semaine dernière et là, impossible de le démarrer !

Il nous a été remonté ce problème dans un contexte de bureau à distance.Le modeleur est dans un état inconsistant.Pour réinitialiser le modeleur veuillez supprimer le répertoire `./workspace` puis relancez-le.

7.2.3. Je ne peux pas créer certains éléments dans l'explorateur de modèle.

Il vous faut vérifier la configuration du prisme en premier lieu.Si le type de l'élément ayant le problème vient d'être ajouté, vous devez fermer et ré-ouvrir l'explorateur de modèle.Si le problème est toujours là, le problème doit surement venir d'une contribution au menu via l'API SmartEA.

7.2.4. Le modeleur est lent et son comportement est anormal (le menu clic droit sur une représentation est vide, ...).

Il y a deux causes connues :

- L'exécutable du modeleur a été renommé. Vous devez restaurer le nom originel de l'exécutable car la configuration par défaut n'est plus prise en compte.
- Le paramètre mémoire RAM du modeleur est trop faible et doit être augmenté. Comment reconnaître ce cas ?

Vous devez comparer la consommation mémoire RAM (1) avec la quantité de mémoire maximum que peut utiliser le modeleur (2) :

- (1) reproduire le comportement anormal et consulter l'outil de monitoring du système d'exploitation pour connaitre la consommation en RAM sur le processus du modeleur.
- (2) Ouvrir le fichier Obeo-SmartEA-Modeler.ini et trouver le paramètre «-Xmx».

-> si (1) est proche de (2) alors il faut augmenter la valeur du paramètre «-Xmx». Par exemple, passez de -Xmx1024m à -Xmx2048m (minimum conseillé).

7.2.5. Linux : le modeleur refuse de démarrer (Problème GTK).

Si vous avez l'erreur suivante :

```
org.eclipse.swt.SWTError: No more handles [Browser style SWT.MOZILLA and Java system property
 org.eclipse.swt.browser.DefaultType=mozilla are not supported with GTK 3 as XULRunner is not
 ported for GTK 3 yet]
```

vous devez:

- laisser `htmlRenderer=default` dans le fichier `application.properties`,
- et exporter la variable d'environnement `SWT_GTK3` avec la valeur `0` (`export SWT_GTK3=0`) avant d'exécuter `./Obeo-SmartEA-Modeler`.

7.2.6. Linux : le modeleur refuse de démarrer (Could not load SWT library)

Si vous avez l'erreur suivante :

```
org.eclipse.swt.SWTError: No more handles [MOZILLA_FIVE_HOME='/usr/lib/mozilla']
 (java.lang.UnsatisfiedLinkError: Could not load SWT library. Reasons:
    /home/...../Obeo-SmartEA-Modeler/configuration/org.eclipse.osgi/693/0/.cp/libswt-
mozilla-gtk-4763.so: libxpcom.so: cannot open shared object file: No such file or directory
    no swt-mozilla-gtk in java.library.path
    Can't load library: /home/....../.swt/lib/linux/x86_64/libswt-mozilla-gtk-4763.so
    Can't load library: /home/....../.swt/lib/linux/x86_64/libswt-mozilla-gtk.so
    ....
```

vérifiez que le package `libwebkitgtk-1.0-0` qui fournit `libxpcom.so` est installé.

Si ce n'est pas le cas, installez ce package avec par exemple la commande

```
sudo apt-get install libwebkitgtk-1.0-0
```

7.2.7. Linux : le modeleur est lent et/ou se ferme brutalement (Problème Cairo).

Si vous avez l'erreur suivante dans la console :

```
eclipse: cairo-misc.c:380: _cairo_operator_bounded_by_source: Assertion `NOT_REACHED' failed.
```

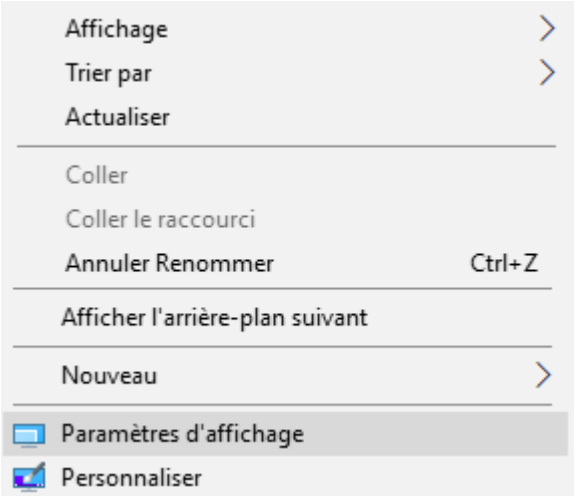
ajoutez la ligne suivante à la fin du fichier `Obeo-SmartEA-Modeler.ini` :

```
-Dorg.eclipse.swt.internal.gtk.cairoGraphics=false
```

7.2.8. En mode édition, les labels des formes graphiques dans les diagrammes sont plus gros (ou petits) sur mon poste que sur les postes de mes collègues.

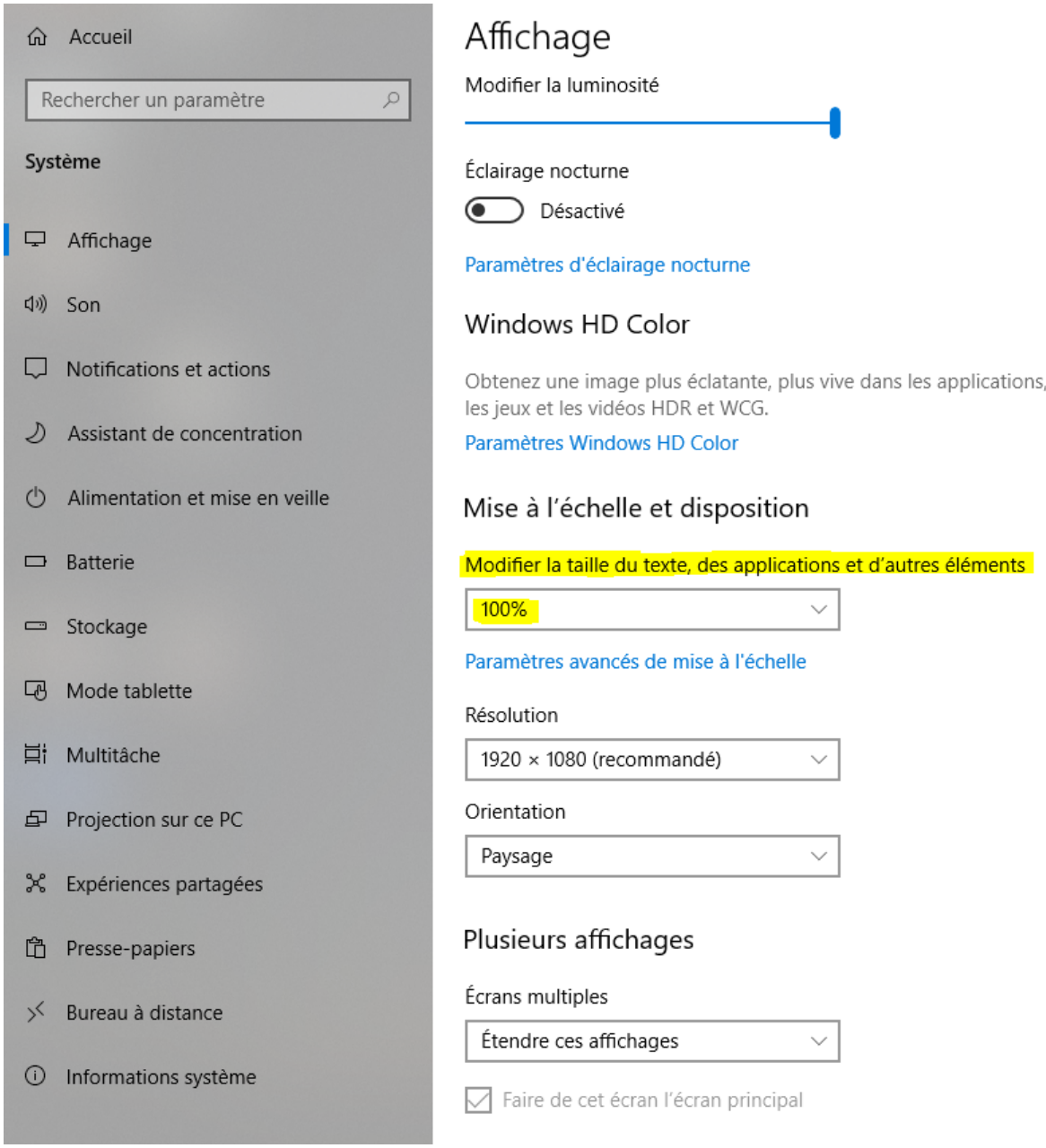
Sous Windows, vérifiez que vous n'avez pas modifié la taille du texte des applications dans les paramètres Windows de mise à l'échelle.

Ouvrez les paramètres d'affichage en faisant clic droit sur le bureau Windows, puis sélectionnez `Paramètres d'Affichage`



Vérifiez ensuite que la taille du texte, des applications et d’autres éléments est exactement à 100%.

Si ce n’est pas le cas, mettez la valeur 100%. Redémarrez ensuite votre modeleur pour que le nouveau paramètre soit pris en compte par le modeleur.



7.2.9. Https : J'ai une erreur SSLHandshakeException au démarrage (sun.security.validator.ValidatorException)

```
javax.net.ssl.SSLHandshakeException: sun.security.validator.ValidatorException: PKIX path
 building failed: sun.security.provider.certpath.SunCertPathBuilderException: unable to find
 valid certification path to requested target
```

La JRE utilisée par le modeleur SmartEA ne peut pas vérifier le certificat SSL du site SmartEA.La solution est la même que pour la FAQServer11.

8. Fichiers de configuration

8.1. Serveur

8.1.1. obeo-smartea-server-licensekey.bat

Permet d'obtenir un registration ID sous Windows afin de faire une demande de licence. Il faut impérativement exécuter ce script **sur la machine** et **avec le compte utilisateur** qui lancera le serveur.

8.1.2. obeo-smartea-server-licensekey.sh

Idem que `obeo-smartea-server-licensekey.bat`, mais pour plateforme Unix.

8.1.3. obeo-smartea-server.bat

Permet de gérer le cycle de vie du serveur (utiliser le .bat pour windows et .sh pour Unix).

Les commandes disponibles sont :

- `start` : Démarrer le serveur
- `status` : Connaître l'état du serveur
- `stop` : Stopper le serveur

8.1.4. obeo-smartea-server.sh

Idem que `obeo-smartea-server.bat` pour plateforme Unix.

8.1.5. obeo-smartea-server-start.bat

Permet de lancer le serveur. Ce script est un raccourci de `obeo-smartea-server.bat start`

8.1.6. obeo-smartea-server-stop.bat

Permet de stopper le serveur. Ce script est un raccourci de `obeo-smartea-server.bat stop`

8.1.7. Obeo-SmartEA-Server.ini

Pour préciser une JVM particulière, ajoutez l'option suivante AVANT l'option `vmargs` :

```
-vm
<path_to_vm>
```

Exemple pour Windows :

```
-vm
C:\Java\JDK\bin\javaw.exe
```

Exemple pour Linux :

```
-vm
/opt/java/bin/java
```

`-Xmx` : (`512M`) Permet de configurer la quantité maximum de mémoire RAM que peut prendre le serveur.

`-Djetty.http.port` : (`8080`) définit le port d'écoute du serveur HTTP

`-Djetty.https.port` : définit le port HTTPS si vous souhaitez utiliser ce mode. Veuillez vous reporter à la section Utiliser le HTTPS pour plus d'information.

`-Dcom.sun.management.jmxremote.port` : (`9999`) port JMX, utilisé par le script d'arrêt du serveur.

`-Dorg.eclipse.net4j.tcp.ssl.passphrase` : pass phrase. Paramètre utilisé pour mettre en place une connexion SSL entre les modeleurs et le serveur CDO.

`-Dorg.eclipse.net4j.tcp.ssl.key` : chemin absolu vers le Key Store. Paramètre utilisé pour mettre en place une connexion SSL entre les modeleurs et le serveur CDO.

Les autres options **ne doivent pas être modifiées**.

8.1.8. etc/application.conf

`include.internaldb` : (`application_internaldb.conf`) Chemin de la configuration de la base de données interne.

`include.cdo` : (`application_cdo.conf`) Chemin de la configuration de CDO.

`include.auth` : (`application_auth.conf`) Chemin de la configuration de l'authentification.

`include.log` : (`application_log.conf`) Chemin de la configuration des traces d'exécution.

`lang` : (`fr` | `en`) définit la locale utilisée par le serveur.

`session.secure` : (`true` | `false`) Restreint l'usage des cookies aux requêtes HTTPS.

`router.automaticRedirection` : (`true` | `false`) définit si l'utilisateur doit être redirigé automatiquement (=true) ou non (=false) vers sa page d'accueil, dans le cas où il essaie d'accéder à un projet, prisme ou branche dont il ne dispose pas des droits (par défaut, l'utilisateur n'est pas redirigé automatiquement).

`index.commitSizeLimit` : (`100`) définit une limite maximale d'évènements à afficher par changement utilisateur pour éviter des problèmes de performances dans l'interface graphique web.

`date.format` : (`yyyy-MM-dd H:mm`) définit le format de date utilisé dans l'interface graphique web.

`projects.order` : (`@, @projet_id2`) définit un ordre particulier dans l'affichage des projets.

`file.upload.maxsize` : (`2000000`) définit la taille maximale des ressources pouvant être téléchargées dans le référentiel (en bytes).

`file.upload.extensions.authorized` : (`*`) Liste des extensions des fichiers pouvant être téléversés. Exemple : pdf,doc,xls,jpg,png,gif

`image.upload.extensions.authorized` : (`*`) Liste des extensions des images pouvant être téléversées. Exemple : jpg,png,gif

`enable.upload.file.content.analysis` : (`false`) Activer l'analyse du contenu des fichiers pour renforcer le filtre des extensions (contôle que le contenu des fichiers uploadés sur le serveur correspond à leurs extensions). La valeur par défaut est false. Voir https://tika.apache.org/2.9.1/formats.html pour les extensions supportées.

`login.logo` : définit l'image d'entête de la page d'authentification. Ne pas définir l'option pour laisser l'image SmartEA. Définir et laisser vide pour supprimer l'image. Si l'image n'est pas dans le répertoire etc/, le chemin indiqué doit être relatif au répertoire etc/ du serveur.

`login.smartea.icon` : (`true` | `false`) Mettre à false pour supprimer l'icône SmartEA sur l'onglet de la page d'authentification.

`login.disclaimer` : définit l'avertissement à ajouter sous le formulaire d'authentification.

`quick.search.wildcards` : (`AUTO`) définit le mode de utilisé par la recherche rapide

`rcp.hosts.authorized` : liste blanche des sites (host) que les modeleurs ont le droit de consulter. Si un utilisateur tente d'accéder à un site n'appartenant pas à cette liste à partir d'un modeleur, alors un message d'erreur indiquera à l'utilisateur que l'accès n'est pas autorisé. Si la liste est vide ou si ce paramètre n'est pas défini alors il n'y a aucune limitation d'accès.

`support.contact` : positionnez la valeur à `true` pour activer la fonctionnalité permettant d'activer un lien en pied de page des pages web, lien permettant à l'utiisateur d'envoyer un email depuis n'importe quelle page web.

`support.contact.email` : permet de renseigner l'adresse email du destinataire.

`support.contact.subject` : permet de renseigner le sujet par défaut de l'email.

`support.contact.body` : permet de renseigner le contenu par défaut de l'email.

Les variables suivantes peuvent être utilisées pour `support.contact.subject` et `support.contact.body` :

- `$location` : l'url de la page courante.
- `$version` : la version de SmartEA.
- `$user` : l'identifiant de l'utilisateur courant.

La variable suivante peut être utilisée pour `support.contact.body`

- `$cr` : retour chariot.

Pour ajouter un texte dans le pied de page des pages web un paramètre est disponible :

- `footer.specific` : texte html qui apparaît dans le pied de page des pages web.

3 paramètres sont disponibles pour personnaliser les tables de résultats des Smart Requests :

- `tables.page.lengthMenu` : Définit les longueurs de page pour la pagination des résultats des Smart Requests (valeurs séparées par une virgule et le tout entre crochets []). Une longueur de -1 correspond à une option d'affichage de tous les résultats. Par exemple [10, 100, -1] indique que les utilisateurs peuvent choisir entre une longueur de page de 10, de 100 ou de ne pas paginer les résultats.
- `tables.page.defaultLength` : Définit la longueur de page par défaut pour les résultats de Smart Requests.
- `tables.filter.threshold` : Par une valeur entre 0 et 1 vous pouvez définir le seuil d'affichage des filtres sur les colonnes des résultats de Smart Requests en fonction de la proportion de valeurs uniques qu'elles contiennent.
  - Un seuil de 0.5 signifie que les filtres s'afficheront si au moins 50% des valeurs de la colonne sont uniques,
  - Un seuil de 1 signifie que les filtres s'afficheront systématiquement,
  - Un seuil de 0 signifie que les filtres ne s'afficheront jamais.

Pour plus de précisions vous pouvez consulter cette page

8.1.9. etc/application_internaldb.conf

`db.url` : définit l'URL JDBC d'accès à la base de données interne SmartEA

`db.driver` : (`org.h2.Driver` | `org.postgresql.Driver`) définit le pilote JDBC à utiliser pour à la base de données interne SmartEA

`db.user` : (`yyyy-MM-dd H:mm`) définit l'utilisateur permettant d'accéder à la base de données interne SmartEA

`db.pass` : définit le mot de passe permettant d'accéder à la base de données interne SmartEA

`db.pool.timeout` : (`10000`) définit le timeout en millisecondes de la base de données interne SmartEA

`db.pool.maxSize` : (`30`) définit la taille maximale du pool de la base de données interne SmartEA

`db.pool.minSize` : (`10`) définit la taille minimale du pool de la base de données interne SmartEA

8.1.9.1. Example PostgreSQL

```
db.url=jdbc:postgresql:smarteainternaldb
db.driver=org.postgresql.Driver
db.user=smarteauser
db.pass=pass123
```

8.1.9.2. Example H2

```
db.url=jdbc:h2:file:data/run/internal/internaldb;DB_CLOSE_ON_EXIT=FALSE;RECOVER=1
db.driver=org.h2.Driver
db.user=sa
db.pass=
```

8.1.10. etc/application_cdo.conf

`cdo.tcpBindingAddress` : (`0.0.0.0`) l'adresse réseau sur laquelle le serveur écoute (par défaut, le serveur écoute sur toutes les adresses, soit `0.0.0.0`)

`cdo.tcpBindingPort` : (`2037`) le port d'écoute du serveur CDO.

`cdo.protocol` : le protocole utilisé pour les flux entre le serveur CDO et les modeleurs ou les SmartEA Clients. Ce paramètre peut prendre la valeur tcp, ssl, ws ou wss. La valeur par défaut est tcp.

`cdo.cacheSize` : (`10000000`) la taille du cache mémoire CDO (`0` désactive le cache)

`cdo.gzippedStreams` : (`false`) si les flux réseaux CDO doivent être zippés, ceci est utile avec une latence réseau supérieure à 1ms.

`cdo.signalProtocolTimeout` : (`30`) le timeout de signal utilisé pour les communications côté serveur (en secondes)

`cdo.ssl` : (`false`) mettre la valeur `true` si une connexion SSL doit être activée entre le serveur CDO et les modeleurs.

8.1.11. etc/application_auth.conf

`userprovider.impl` : (`fr.obeo.smartea.core.server.impl.user.LdapUserProviderImpl`) Par défaut SmartEA gère l'authentification avec un fichier nommé `users.yml`.Pour activer l'authentification LDAP, il faut utiliser l'implémentation `fr.obeo.smartea.core.server.impl.user.LdapUserProviderImpl`.Il est aussi possible de créer une nouvelle implémentation ad hoc de `fr.obeo.smartea.core.server.api.IUserProviderExtension`.

`userprovider.ldap.url` : (`ldap://ldap.obeo.fr`) l'URL permettant de se connecter au LDAP

`userprovider.ldap.appli.dn` : le `distinguisehd name` (DN) de l'utilisateur applicatif devant être utilisé pour se connecter au LDAP.

`userprovider.ldap.appli.password` : le mot de passe de l'utilisateur applicatif

`userprovider.ldap.baseDN` : le pattern à utiliser pour récupérer le noeud racine contenant les `distinguished names` (DN) des utilisateurs SmartEA (ex : dc=obeo,dc=fr).

`userprovider.ldap.scope` :

- object : si le `distinguished name` (DN) de l'utilisateur SmartEA doit être recherché dans le noeud `userprovider.ldap.baseDN`.
- oneLevel : si le `distinguished name` (DN) de l'utilisateur SmartEA doit être recherché dans le noeud `userprovider.ldap.baseDN` et les noeuds fils de premier niveau.
- subTree : si le `distinguished name` (DN) de l'utilisateur SmartEA doit être recherché dans le noeud `userprovider.ldap.baseDN` et tous les noeuds fils quelle que soit la profondeur de l'arbre.

`userprovider.ldap.userSearchAttribute` : L'attribut à utiliser pour rechercher le `distinguished name` (DN) de l'utilisateur. La valeur par défaut est `uid`. Dans le cas d'Active Directory, vous pouvez utiliser l'attribut `sAMAccountName`.

`userprovider.ldap.firstNameAttribute` : le nom de l'attribut LDAP contenant le prénom de l'utilisateur

`userprovider.ldap.lastNameAttribute` : le nom de l'attribut LDAP contenant le nom de l'utilisateur

`userprovider.ldap.group.baseDN` : le pattern à utiliser pour récupérer le noeud racine contenant les groupes (ex : ou=Group,dc=obeo,dc=fr).

`userprovider.ldap.groupSearchPattern` : Le pattern pour retrouver un utilisateur particulier dans les différents groupes. (ex LDAP avec DN de l'utilisateur : (&(cn=*)(memberUid=USER_DN)), ex LDAP avec identifiant de l'utilisateur : (&(cn=*)(memberUid=LOGIN)), ex AD avec DN de l'utilisateur : (&(cn=*)(member=USER_DN)))

`userprovider.sso.impl` : Pour activer l'authentification SSO, il faut préciser une implémentation de `fr.obeo.smartea.core.server.api.ISSOUserProviderExtension`. Une implémentation OpenID Connect est disponible `fr.obeo.smartea.core.server.impl.user.OpenIdUserProviderImpl` et nécéssite une configuration. A titre d'exemple, nous prendrons une configuration de l'outil Open Source Keycloak https://www.keycloak.org/

`userprovider.openid.discoveryEndpoint` : l'URL «discovery» de votre service OpenID (exemple Keycloak `https://keycloak.obeo.fr:8443/auth/realms/smartea/.well-known/openid-configuration`)

`userprovider.openid.clientId` : L'identifiant de votre client dédié à SmartEA dans votre service OpenID (exemple Keycloak `openid-client`)

`userprovider.openid.clientSecret` : Le «secret» de votre client dédié à SmartEA dans votre service OpenID (exemple Keycloak `d59d8659-2917-4443-be47-8173c2f88410`)

`userprovider.openid.callbackUrl` : L'URL de «callback» vers SmartEA respectant la forme [baseurl]/smartea/sso/callback (exemple Keycloak `http://localhost:8080/smartea/sso/callback`)

`userprovider.openid.useIdTokenExpiration` : (`false` si non précisé) Utilisez l'expiration `IdToken` au lieu de l'expiration `AccessToken`.

`userprovider.openid.scope` : (`openid profile` si non précisé) Le «scope» OpenID à exiger.

`userprovider.openid.responseType` : (`code` si non précisé) Le «response type» à exiger. SmartEA ne gère que le «code flow» OpenID.

`userprovider.openid.acrValues` : (vide si non précisé) Les acrValues à exiger.

`userprovider.openid.leewayWindow` : (`5` si non précisé) la fenêtre de tolérance (en seconde) dans laquelle le jeton doit toujours être considéré comme valide

`userprovider.openid.firstNameAttribute` : (`given_name` si non précisé) clé de l'attribut prénom à récupérer dans `userinfoEndpoint`.

`userprovider.openid.lastNameAttribute` : (`family_name` si non précisé) clé de l'attribut nom à récupérer dans `userinfoEndpoint`.

`userprovider.openid.loginAttribute` : (`sub` si non précisé) clé de l'attribut identifiant à récupérer dans `userinfoEndpoint`.

`userprovider.openid.rolesAttribute` : nom de l'attribut contenant les rôles utilisateurs (à utiliser dans le cas d'utilisation des groupes).

`userprovider.openid.rolesAsObjects.keyAttribute` : si le serveur OpenID ne renvoie pas les rôles sous formes de simples chaînes de caractères mais d'objets, cet attribut permet de spécifier la clef des objets à utiliser pour obtenir l'identifiant des rôles.

`useraccessmanager.impl` : l'implémentation du gestionnaire de profils. Nous proposons `fr.obeo.smartea.core.server.internal.user.YAMLBasedUserAccessManagerImpl` pour relier les utilisateurs d'un LDAP à des prismes au travers d'un fichier de configuration. Nous proposons également :

- `useraccessmanager.impl=fr.obeo.smartea.core.server.impl.user.YAMLBasedUserAndGroupAccessManagerIm` pour relier les utilisateurs ET les groupes d'un LDAP à des prismes au travers d'un fichier de configuration.

- `useraccessmanager.impl=fr.obeo.smartea.core.server.impl.user.YAMLBasedSSOUserAndGroupAccessManage` pour relier les utilisateurs ET les groupes d'un SSO à des prismes au travers d'un fichier de configuration.

`useraccessmanager.yamlbased.userProfilesResource` : le chemin du fichier contenant les associations entre utilisateurs et prismes (par défaut `etc/profiles.yml`), se reporter à la section profiles.yml pour plus d'informations.

8.1.12. etc/application_log.conf

Ce fichier contient le paramétrage des traces applicatives générées par le serveur SmartEA. C'est un fichier de configuration log4j. Se reporter à la documentation du produit pour plus d'informations.

Ce paramétrage n'est cependant pas censé être modifié.

`log4j.rootLogger` : Configuration de trace par défaut.

`log4j.logger.fr.obeo.smartea` : Configuration de trace du code SmartEA.

8.1.13. etc/users.yml

Ce fichier est utilisé par défaut lorsque le mode LDAP n'est pas activé. Il contient la déclaration de l'ensemble des utilisateurs pouvant accéder au référentiel.

Attention à bien respecter la syntaxe YML. Les espaces et le retour à la ligne ont une importance.

Exemple :

```
afontaine:
    password: 123
    firstName: Alexandre
    lastName: Fontaine
    admin: true
    prismIDs: {
        VoyageDiscount: [EnterpriseArchitect, CIO, UnitManager],
        Sandbox: [EnterpriseArchitect, CIO, UnitManager],
        Public
    }
```

Le paramétrage est le suivant :

- `afontaine` : définit l'identifiant de l'utilisateur
  - `password` : définit le mot de passe de l'utilisateur
  - `firstName` : définit le prénom de l'utilisateur
  - `lastName` : définit le nom de l'utilisateur
  - `admin`: définit l'accès à l'administration
  - `prismIDs` : définit les accès permis à l'utilisateur par projet et prisme, dans l'exemple ci-dessus :
    - `VoyageDiscount: [EnterpriseArchitect, CIO, UnitManager]` : indique que l'utilisateur a accès aux prismes `EnterpriseArchitect`, `CIO` et `UnitManager` du projet `VoyageDiscount`.
    - `Public` : indique que l'utilisateur a accès au prisme `Public` sur n'importe quel projet.

8.1.14. etc/profiles.yml

Ce fichier contient les associations entre les identifiants des utilisateurs et ceux des prismes auxquels ils sont rattachés.

Une directive permet en complément de définir les prismes accessibles par défaut pour un utilisateur authentifié auprès du LDAP/ActiveDirectory/SSO mais non déclaré dans le fichier.

Attention à bien respecter la syntaxe YML. Les espaces et le retour à la ligne ont une importance.

Exemple :

```
$DEFAULT$ :
    prismIDs: { Public }
```

Il est également possible de définir des associations entre les identifiants des groupes de l'annuaire (non supporté en SSO) et ceux des prismes auxquels ils sont rattachés.

Pour cela, il suffit d'ajouter l'attribut `group: true`.

```
commercial:
    group: true
    prismIDs: {
        sandbox: [CommerceLS],
        voyagediscount: [CommerceLS]
    }
```

Il est inutile de redémarrer le serveur lorsque ce fichier est modifié. Les modifications qui y sont apportées sont prises en charge à chaud. Par contre les modifications apportées pour un utilisateur donné ne sont appliquées que lorsque celui se reconnecte (si il était connecté au moment des modifications).

8.1.15. etc/service-users.yml

En mode SSO, la connexion des utilisateurs se fait par une redirection sur un site tiers.Cet enchaînement n'est pas adapté pour un usage de type client ou modeleur de publication":#PublicationModeler (compte de service).En effet, les clients ou les modeleurs de publication doivent pouvoir être démarrés automatiquement sans intervention humaine.

Le fichier `service-users.yml` répond à ce besoin en listant les identifiants autorisés à cet usage.

Ce fichier est uniquement utilisé en mode SSO.

```
client1:
    password: 123
client2:
    password: 123
```

En mode LDAP ou Active Directory (utilisation du user provider `userprovider.impl=fr.obeo.smartea.core.server.impl.user.LdapServiceUserProviderImpl`) vous n'avez pas à déclarer les mots de passe. L'authentification est faite par votre annuaire.

8.1.16. etc/smartEALicense.key

Ce fichier contient les informations de licence pour démarrer SmartEA.

8.1.17. etc/projects/{projet_id}.conf

`label` : le libellé.

`description` : la description.

`enabled` : true si le projet est actif, false sinon.

`sql.request.maxLogicalIdsRetrievedByRequest` : (`256`) le nombre maximum d'identifiants logiques pouvant être demandés par requête SQL.

`db.xxx` : le paramétrage d'accès à la base de données du projet associé (paramétrage identique à celui dédié au schéma de base de données interne SmartEA, voir la section application_internaldb.conf pour plus d'informations)

`dependency_graph.item_by_relation.max` : (`14`) le nombre maximum d'éléments à afficher par relation dans le diagramme de dépendance.

`artifacts.sirius.automaticCreation` : la liste des représentations Sirius à créer automatiquement suite à la création d'un objet sémantique dont le type est racine de la représentation. Par exemple, viewpointId=ArchimateAddOn,representationId=archimate.ApplicationFlowDiagram

`default.path.project` : (`defaults/Default`) le chemin vers un export de projet dézippé à utiliser pour initialiser le projet.

`default.path.prism` : (`defaults/default.prism`) le chemin du fichier contenant le modèle de prismes par défaut à utiliser.

`default.path.semantic` : (`defaults/default.semantic`) le chemin du fichier contenant le modèle sémantique par défaut à utiliser.

`default.path.preferences` : (`defaults/filename.preferences`) le chemin du fichier contenant le modèle de préférences par défaut à utiliser.

`page.header.scale` : Échelle de l'en-tête des pages web des artefacts. Pour diminuer la hauteur de l'en-tête définissez une valeur inférieure à 1, `0.7` par exemple.

8.1.18. etc/projects/{projet_id}.publication

La publication sur commit permet de déclencher la publication d'un ensemble de représentations Sirius suite à un commit sémantique.

8.1.19. Gestion des préférences projet partagées

Chaque projet dispose d'un ensemble de préférences partagées entre tous les utilisateurs. Ces préférences sont supposées être éditées dans le modeleur à l'aide du dialogue dédié (accessible par un bouton dans la barre d'outils du modeleur).

Pour contribuer des pages de préférences, veuillez vous référer au chapitre «Préférences projet» du guide de personnalisation.

Une fois ces pages créées et déployées, il est possible de définir des valeurs par défaut pour ces préférences depuis le serveur dans le répertoire `data/defaults` au travers des ressources suivantes :

- `default.preferences` : les préférences par défaut globales
- `<project_id>.preferences` : les préférences par défaut pour un projet donné (identifié par son identifiant, exemple : `Sandbox.preferences`)

Ces ressources sont de simples fichiers de propriétés, chaque valeur étant associée au chemin de la préférence associée.

Par exemple : `smarteacore/automaticDiagramImagePublishing=true` définit la préférence par défaut qui est présente dans la page `smarteacore` et dont le code est `automaticDiagramImagePublishing`.

Si vous éditez ces fichiers, inutile de redémarrer le serveur, les mises à jour sont prises en compte à chaud.

Note additionnelle : avoir une page de préférence pour une préférence donnée n'est pas obligatoire. Dit autrement, si vous configurez une valeur par défaut côté serveur pour une préférence donnée, même si aucune page de préférence n'existe pour cette préférence, celle-ci sera quand même utilisable. Cette astuce peut permettre de définir une préférence qui peut n'être pilotée que côté serveur.

8.2. Modeleur

8.2.1. application.properties

`https` : (`true`) Permet d'activer la connexion https du Modeleur.

`cdo.protocol` : Permet de préciser le protocole utilisé entre le serveur CDO et le modeleur. Ce paramètre peut prendre la valeur tcp, ssl, ws ou wss. La valeur par défaut est tcp.

`webServer` : (`localhost`) Permet de configurer l'hôte du serveur web.

`webPort` : (`8080`) Permet de configurer Le port du serveur web.

`htmlRenderer` : (`edge`) Permet de définir le moteur de rendu à utiliser. Edge doit être utilisé sous Windows (Microsoft WebView2). Sous Linux il est possible d'utiliser Webkit ou le navigateur web par défaut du système d'exploitation.

`cdoSessionTimeout` : (`30`) Permet de définir le «timeout» pour se connecter au serveur CDO.

`debug` : (`true`) Permet d'activer d'avantage de traces d'exécution.

`closeViews` : (`true`) Permet de configurer le modeleur pour qu'il ne ferme pas les vues Propriétés et Journal d'erreurs lors de la sauvegarde et lors de l'ouverture des représentations en mode édition.

8.2.2. Obeo-SmartEA-Modeler.ini

Pour préciser une JVM particulière, ajoutez l'option suivante AVANT l'option `vmargs` :

```
-vm
<path_to_vm>
```

Exemple pour Windows :

```
-vm
C:\Java\JDK\bin\javaw.exe
```

Exemple pour Linux :

```
-vm
/opt/Java/bin/java
```

`-Xmx` : (`2048m`) Permet de configurer la quantité maximum de mémoire RAM que peut utiliser le modeleur.

`-Duser.language` : (`fr`) Permet de configurer la langue de l'interface graphique du modeleur.

`-Dfr.obeo.smartea.core.rcp.ui.session.internal.feedback.`

`DisplayRemovedObjectsModelChangeTrigger.priority` : (`0`) La priorité du trigger affichant une boîte de dialogue demandant confirmation d'une suppression d'un objet sémantique.

`-Dfr.obeo.smartea.core.rcp.ui.ignoreAIRDDuringTreeTableAndMatriceOpening` : (`true` ou `false` si non précisé) Permet de préciser si les arbres, tables et matrices doivent être recréés à chaque ouverture.

`-Dfr.obeo.smartea.core.rcp.update.showInformation` :

- `true` (valeur par défaut) : Le popup indiquant qu'une mise à jour du modeleur est nécessaire affiche les plugins qui vont être installés et propose un bouton permettant d'annuler la mise à jour.
- `false` : Le popup indiquant qu'une mise à jour du modeleur est nécessaire informe qu'une mise à jour doit être faite sans plus d'information et ne permet pas d'annuler l'opération.

`-Dorg.eclipse.net4j.tcp.ssl.passphrase` : pass phrase. Paramètre utilisé pour mettre en place une connexion SSL entre le modeleur et le serveur CDO.

`-Dorg.eclipse.net4j.tcp.ssl.trust` : Chemin absolu vers le Trust Store. Paramètre utilisé pour mettre en place une connexion SSL entre le modeleur et le serveur CDO.

`-Dorg.eclipse.swt.browser.EdgeDir` : Chemin vers le répertoire contenant WebView2 dézippé.

`-Dfr.obeo.smartea.archimate.meff.exchanger.importer.routing.style` : (`rectilinear` ou `oblique` si non précisé) Permet de préciser le type des liens des relations ArchiMate dans les diagrammes lors d'un import au format d'échange ArchiMate.

`-Dfr.obeo.smartea.archimate.meff.exchanger.importer.routing.closestdistance` : (`true` si non précisé) Permet de préciser la valeur de la propriété `closest distance` des liens des relations ArchiMate dans les diagrammes lors d'un import au format d'échange ArchiMate.

`-Dfr.obeo.smartea.archimate.meff.exchanger.importer.routing.avoidobstructions` : (`true` si non précisé) Permet de préciser la valeur de la propriété `avoid obstruction` des liens des relations ArchiMate dans les diagrammes lors d'un import au format d'échange ArchiMate.

Les autres options **ne doivent pas être modifiées**.