SG/SPSSI/CPII
DOSO/PNE Sécurité

RAPPORT

09/03/2017

Portail Cerbère d'authentification et d'autorisations

Installation de la librairie Cerbère bouchon.

Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

MINISTÈRE
DE L'ÉGALITÉ DES TERRITOIRES
ET DU LOGEMENT
www.territoires.gouv.fr

MINISTÈRE DE L'ÉCOLOGIE,
DU DÉVELOPPEMENT DURABLE
ET DE L'ÉNERGIE
www.developpement-durable.gouv.fr

Historique des versions du document

Date	Commentaire
09/03/2017	Pré-requis Java 1.7 à compter de la version 4.3.0
25/08/2014	Attributs Secure et HttpOnly sur cookie Cerbère
28/08/2013	Précision sur l'obligation d'appartenance de chaque utilisateur à un service. Remise à la charte.
03/05/2012	Intégration des entreprises et professionnels.
04/08/2011	Remise en forme.

Affaire suivie par

PNE Sécurité - SG/SPSSI/CPII – PNE Sécurité
Courriel : pne-securite@developpement-durable.gouv.fr

Rédacteur

Erwan SALMON – SG/SPSSI/CPII – PNE Sécurité
Patrick MARGUINAL – SG/SPSSI/CPII – PNE Sécurité

Relecteurs

SOMMAIRE

1 - Présentation

1.1 - Contenu de la librairie Cerbère bouchon

Le paquetage Cerbère-bouchon contient l'ensemble des APIs Cerbère, permettant la gestion de l'authentification et des droits des utilisateurs sur les applications web. Ces APIs permettent le développement d'application web utilisant Cerbère pour l'authentification et la gestion des droits. Cerbère-bouchon permet de développer un site sans pour autant avoir Cerbère en ligne.

L'interface de programmation (API) Cerbère permet aux applications du ministère du développement-durable d'interagir avec le système de sécurité au moyen de méthodes Java. Elle permet essentiellement :

- d'obtenir des informations sur l'utilisateur connecté (carte de visite) ;
- d'obtenir des informations sur l'entreprise à laquelle appartient l'utilisateur;
- d'obtenir des informations sur les droits de cet utilisateur (profil applicatif);
- d'agir sur l'utilisateur connecté (ré-authentification ou deconnexion).

Ce document décrit les différentes classes de cet interface de programmation.

1.2 - Principe de fonctionnement général du filtre Cerbère

Le filtre Cerbère garantit à cette application que tout utilisateur a été au préalable authentifié selon le niveau requis par cette application.
Il intercepte toute requête sur l'application (ou partie de l'application selon la configuration retenue) et ramène l'utilisateur vers le portail d'authentification Cerbère s'il n'est pas authentifié ou si ses autorisations sont périmées.
Le filtre Cerbère contient également les interfaces de programmation qui permettent aux application d'avoir accès aux informations souhaitées sur l'utilisateur (identité, autorisations, ..).

Dans le cas du mode bouchon le portail d'authentification devient une simple interface d'authentification embarquée dans la librairie afin d'en simplifier son déploiement.

2 - Installation

2.1 - Pré-requis

Les pré-requis d'installation du filtre Cerbère en version 4 sont les suivants :

- Une machine virtuelle Java de version **1.7** ou plus.
- Un environnement serveur Java (servlet 2.3 ou plus).

2.2 - Installation librairies

Le filtre Cerbère est fourni par la librairie `cerbere-bouchon-xyz.jar`.
Aucune dépendance n'est requise.

Procédure : copier dans le dossier `WEB-INF/lib` de l'application la librairie `cerbere-filtre-xyz.jar`.

2.3 - Configuration du filtre

Le filtre Cerbère est géré par la classe `i2.application.cerbere.filtre.FiltreCerbere`.
La portée du filtre Cerbère (élement `<filter-mapping>`) dépend de l'application, l'exemple ci-dessous considère que toute l'application est sécurisée par Cerbère

```
<!-- Définition du filtre Cerbère. -->
<filter>
  <filter-name>FiltreCerbere</filter-name>
  <filter-class>i2.[...].FiltreCerbere</filter-class>
  <!-- paramètres d'initialisation -->
  ....
</filter>

<!-- Sécurisation de toute l'application. -->
<filter-mapping>
  <filter-name>FiltreCerbere</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

Le comportement du filtre peut être modifié selon les paramètres d'initialisation décrits ci-dessous.

2.3.1 - Paramètre : identifiant de l'application

Ce paramètre est obligatoire (tous les autres possèdent des valeurs par défaut).

Cerbère-bouchon utilise un référentiel d'utilisateurs et d'applications contenu dans un fichier XML. L'application à sécuriser est définie dans ce fichier XML et y possède un identifiant. Cet identifiant **doit** être fourni au filtre pour son initialisation.
Pour plus d'information sur le format du fichier référentiel XML, se rapporter aux chapitres suivants.

Syntaxe :
Nom du paramètre : applicationId
valeur : identifiant de l'application tel que défini dans le fichier référentiel XML
défaut : aucune valeur par défaut

Exemple :

```
<!-- définition du filtre Cerbère -->
<filter>
```

```
  <filter-name>FiltreCerbere</filter-name>
  <filter-class>i2.[...].FiltreCerbere</filter-class>
  <!-- paramètres d'initialisation -->
  ...
  <!-- identifiant (***OBLIGATOIRE***) de l'application -->
  <init-param>
    <param-name>applicationId</param-name>
    <param-value>restaurant</param-value>
  </init-param>
    ...
</filter>
```

2.3.2 - Paramètre : fichier référentiel XML

Ce paramètre est facultatif.

Le filtre doit savoir où trouver le fichier référentiel XML. Par défaut, le filtre recherche un fichier nommé `cerbere-filtre-bouchon.xml` sous WEB-INF ou dans le classpath. L'emplacement ou le nom du fichier peut être modifié par le paramètre `conf`. Le fichier indiqué dans ce paramètre sera recherché en plusieurs endroits, sitôt le fichier trouvé, l'algorithme de recherche s'arrête. L'algorithme de recherche du ficher est le suivant :

- le nom de fichier est supposé absolu(à partir de la racine du système de fichier), le filtre regarde si un fichier existe sur disque avec le nom indiqué (java.util.File.isFile(conf));
- si le fichier n'est pas trouvé, alors recherche en relatif sous `WEB-INF/`;
- si le fichier n'est pas trouvé, alors recherche dans le classpath

Syntaxe :
Nom du paramètre : `conf`
valeur : chemin complet du fichier référentiel XML, ou relatif sous `WEB-INF/`, ou dans le classpath
défaut : `cerbere-filtre-bouchon.xml`

Exemple :
```
<!-- définition du filtre Cerbère -->
<filter>
  <filter-name>FiltreCerbere</filter-name>
  <filter-class>i2.[...].FiltreCerbere</filter-class>
  <!-- paramètres d'initialisation -->
  ...
  <!-- emplacement du fichier référentiel XML -->
  <init-param>
    <param-name>conf</param-name>
    <param-value>cactus-filtre.xml</param-value>
  </init-param>
    ...
</filter>
```

2.3.3 - Paramètre : point d'entrée dans l'application

Ce paramètre est facultatif.

Le filtre peut forcer l'entrée dans l'application sur une URL donnée. Par défaut, l'entrée dans l'application se fait à l'URL indiquée par l'utilisateur ou à l'URL imposée par la logique interne à l'application.
Pour que Cerbère gère l'adresse d'entrée dans l'application, il faut passer au filtre le paramètre d'initialisation `applicationEntree`.

Syntaxe :
Nom du paramètre : `applicationEntree`
valeur : URL interne à l'application
défaut : vide

Exemple :

```
<!-- définition du filtre Cerbère -->
<filter>
  <filter-name>FiltreCerbere</filter-name>
  <filter-class>i2.[...].FiltreCerbere</filter-class>
  <!-- paramètres d'initialisation -->
  ...
  <!-- point d'entrée dans l'aplication -->
  <init-param>
    <param-name>applicationEntree</param-name>
    <param-value>/accueil.do</param-value>
  </init-param>
  ...
</filter>
```

2.3.4 - Paramètre : traces du filtre

Ce paramètre est facultatif.

Le filtre Cerbère-bouchon peut tracer ses actions. Deux paramètres sont définis pour cela :

- `log.niveau` : niveau de trace
- `log.fichier` : emplacement du fichier de trace, ou `stdout` pour sortie standard

Syntaxe :

Nom du paramètre : `log.niveau`
valeur : `OFF` | `INFO` | `CONFIG` | `DEBUG` | `DEV`
défaut : `ERROR`

Nom du paramètre : `log.fichier`
valeur : emplacement du fichier de traces, ou `stdout` pour sortie standard
défaut : `stdout`

Exemple :

```
<!-- définition du filtre Cerbère -->
<filter>
  <filter-name>FiltreCerbere</filter-name>
  <filter-class>i2.[...].FiltreCerbere</filter-class>
  <!-- paramètres d'initialisation -->
  ...
  <!-- initialisation des traces -->
  <init-param>
    <param-name>log.niveau</param-name>
    <param-value>info</param-value>
  </init-param>
  <init-param>
    <param-name>log.fichier</param-name>
    <param-value>/var/log/cerbere/filtre.log</param-value>
  </init-param>
  ...
</filter>
```

2.3.5 - Paramètre : URLs non sécurisées

Ce paramètre est facultatif.

Le filtre Cerbère peut ignorer certaines URL. Cela permet par exemple de superviser des URL de l'application sans devoir s'authentifier ou de définir une portée du filtre de type "tous sauf ...".
Attention, les URLs concernées par ce filtre ne seront pas sécurisées, elles ne doivent pas donner accès à de l'information confidentielle.

Les URLs non sécurisées sont définies par une suite d'expression délimitées par des points-virgule ';'. Si une URL utilisateur correspond à l'une de ces expressions, le filtre laissera passer la requête sans s'assurer de l'identité ni des droits de l'utilisateur.
Les expressions peuvent être simples (*.gif, /public/*), elles peuvent aussi être régulières si préfixées par `re:` ou `re:i:`. La syntaxe des expressions régulières est celle définie par la classe java.util.regex.Pattern (voir javadoc correspondante).

Il existe deux motifs pré-définis permettant de lister la plupart des ressources de type images, style et javascript. Ces deux motifs peuvent être utilisés pour lister les fichiers d'extension .gif; .jpeg; .jpg; .png; .css et .js :

- `#MEDIA` : liste des extensions .gif; .jpeg; .jpg; .png; .css; .js
- `#MEDIA_RE` : liste des extensions .gif; .jpeg; .jpg; .png; .css; .js sous forme d'expression rationnelle, à utiliser dans une liste de la forme `re:#MEDIA_RE;<autres_urls>`.

Exemple :

- `/__supervision__/*` : tous les URLS de type <*webapp*>/__supervision__/*
- `re:/__supervision__/.*` : tous les URLS de type <*webapp*>/__supervision__/*
- `re:i:/__supervision__/.*` : tous les URLS de type <*webapp*>/__supervision__/*, sans distinction de casse.

Note : un utilitaire en ligne de commande est fourni dans la livraison Cerbère-bouchon, il s'agit du script `RegExpTest`(`.bat`|`.sh`) présent sous `bin/`. Ce script permet de tester une expression régulière sur une chaîne de caractères données. Sa syntaxe est :

- `RegExpTest.bat -e <expression> -t <texte>` sous Windows
- `RegExpTest.sh -e <expression> -t <texte>` sous Linux

Avec
<*expression*> : expression régulière à tester
<*texte*> : texte (URL) à valider sur l'expression régulière
Exemples :
`RegExpTest -e (.*)\.gif -t /images/logo.gif` => "correspondance OK"
`RegExpTest -e (.*)\.gif -t /accueil.do` => "PAS de correspondance"

Syntaxe :
Nom du paramètre : `urls-ouvertes`
valeur : liste d'expressions séparées par des points-virgules ';', la liste doit être préfixée de "`re:`" si expressions régulières sensibles à la casse; de "`re:i:`" si expressions régulières insensibles à la casse.
défaut : `re:i:#MEDIA_RE` <extensions .gif; .jpeg; .jpg; .png; .css; .js>

Exemple :

```
<!-- définition du filtre Cerbère -->
<filter>
  <filter-name>FiltreCerbere</filter-name>
  <filter-class>i2.[...].FiltreCerbere</filter-class>
  <!-- paramètres d'initialisation -->
  ...
  <!-- URLs non sécurisées pour la supervision et les images -->
  <init-param>
    <param-name>urls-ouvertes</param-name>
    <param-value>#MEDIA;/__supervision__/*</param-value>
  </init-param>
  ...
</filter>
```

2.3.6 - Paramètre : Prévention des caches

Ce paramètre est facultatif.

Les serveurs proxy situés entre l'utilisateur et l'application peuvent mettre en cache les documents issus de l'application si ces documents l'autorisent (date de création ou d'expiration présente dans

les en-têtes du document par exemple). Ce comportement est indépendant de Cerbère et dépend de l'application et du serveur qui l'héberge.
Par défaut, Cerbère ne modifie pas ce comportement, il peut sur demande interdire la mise en cache de tous les documents sécurisées en ajoutant automatiquement les en-têtes adéquates dans les réponses HTTP. Les en-têtes `Pragma`, `Cache-Control` et Expire seront ajoutées à toutes les ressources :

- sécurisées par Cerbère;
- ne contenant pas déjà d'en-tête `Pragma`, `Cache-Control` ,Expire ou `Last-Modified`.

Syntaxe :
Nom du paramètre : `cache-directives`
valeur : oui pour interdit la mise en cache, autre valeur sinon.
défaut : `non`

Exemple :
```
<!-- définition du filtre Cerbère -->
<filter>
  <filter-name>FiltreCerbere</filter-name>
  <filter-class>i2.[...].FiltreCerbere</filter-class>
  <!-- paramètres d'initialisation -->
  ...
  <!-- interdir la mise en cache -->
  <init-param>
    <param-name>cache-directives</param-name>
    <param-value>oui</param-value>
  </init-param>
  ...
</filter>
```

2.3.7 - Paramètre : Cache de validation du jeton Cerbère

Ce paramètre est facultatif.

Afin d'optimiser les temps de réponse des applications, le filtre Cerbère cache les jetons de session Cerbère les plus fréquemment reçus pour ne pas avoir à les vérifier à chaque requête. Ce cache est configurable en durée et en taille.

Syntaxe :
Nom du paramètre : `cache.taille`
valeur : taille du cache contenant les jetons de session Cerbère les plus récents.
défaut : 50

Nom du paramètre : `cache.duree`
valeur : durée de validité, en secondes, d'un jeton Cerbère dans le cache
défaut : 300

Exemple :
```
<!-- définition du filtre Cerbère -->
<filter>
  <filter-name>FiltreCerbere</filter-name>
  <filter-class>i2.[...].FiltreCerbere</filter-class>
  <!-- paramètres d'initialisation -->
  ...
  <!-- cache de jetons Cerbère -->
  <init-param>
    <param-name>cache.taille</param-name>
    <param-value>100</param-value>
  </init-param>
  <init-param>
    <param-name>cache.duree</param-name>
    <param-value>100</param-value>
```

```
  </init-param>
  ...
</filter>
```

2.3.8 - Paramètre : clé de hash

Ce paramètre est facultatif.

Le jeton Cerbère-bouchon utilise une clé arbitraire pour générer une empreinte des jetons de session. Cette clé est calculée automatiquement au démarrage du filtre mais peut aussi est spécifiée manuellement.

Syntaxe :
Nom du paramètre : `cle`
valeur : chaîne de caractères quelconque
défaut : <*vide*> (calculée par l'application)

Exemple :
```
<!-- définition du filtre Cerbère -->
<filter>
  <filter-name>FiltreCerbere</filter-name>
  <filter-class>i2.[...].FiltreCerbere</filter-class>
  <!-- paramètres d'initialisation -->
  ...
  <!-- clé pour empreinte -->
  <init-param>
    <param-name>cle</param-name>
    <param-value>abc123</param-value>
  </init-param>
  ...
</filter>
```

2.3.9 - Paramètre : type de challenge

Ce paramètre est facultatif.

L'authentification utilisateur est par défaut faite par mécanisme de stimulation/réponse qui permet de ne pas transmettre en clair le mot de passe utilisateur. Ce mécanisme est sans effet lorsque l'interpréteur Javascript est désactivé.
Par défaut le challenge généré est à usage unique, ce qui convient dans la quasi totalité des cas mais pose problème pour les tests en cluster sur plusieurs serveurs d'applications actifs simultanément. Il est possible de générer un challenge simplifié compatible avec un environnement en cluster.

Syntaxe :
Nom du paramètre : `challenge.type`
valeur : 1 pour challenge fort, 2 pour challenge simplifié pour clusters
défaut : 1

Exemple :
```
<!-- définition du filtre Cerbère -->
<filter>
  <filter-name>FiltreCerbere</filter-name>
  <filter-class>i2.[...].FiltreCerbere</filter-class>
  <!-- paramètres d'initialisation -->
  ...
  <!-- type de challenge -->
  <init-param>
    <param-name>challenge.type</param-name>
    <param-value>2</param-value>
```

```
  </init-param>
  ...
</filter>
```

2.3.10 - Paramètre : jeton de session Cerbère sur protocole TLS uniquement

Ce paramètre est **facultatif**.
Lorsque l'application n'est accessible que dans un canal sécurisé (protocole TLS, ex SSL), il est préférable d'interdire la transmission par le navigateur du jeton de session Cerbère dans un canal non sécurisé (non TLS). Cette protection permet de diminuer le risque de vol de cookie, elle est mise en œuvre par l'ajout de l'attribut "`Secure`" au cookie Cerbère.
Lorsque l'échange sécurisé TLS est direct entre le navigateur et l'application (pas de reverse-proxy), cet attribut "`Secure`" est automatiquement et obligatoirement ajouté au cookie Cerbère.
Lorsque l'échange sécurisé TLS s'arrête sur un reverse-proxy, cet attribut n'est pas défaut pas positionné (l'application n'a pas connaissance du canal chiffré précédent le reverse-proxy). Ce paramètre du filtre Cerbère permet alors d'ajouter l'attribut `Secure` au cookie Cerbère.

Syntaxe :
Nom du paramètre : `session.secure`
valeur : `oui`|`1` pour forcer l'ajout de l'attribut `Secure` au cookie Cerbère.
défaut : `oui`|`1` si application accessible en TLS sans reverse-proxy, `non`|`0` sinon

Exemple :
```
<!-- Définition du filtre Cerbère. -->
<filter>
  <filter-name>FiltreCerbere</filter-name>
  <filter-class>i2.[...].FiltreCerbere</filter-class>
    <!-- Paramètres d'initialisation. -->
    ...
    <!-- Ajout de l'attribut Secure au cookie Cerbère -->
  <init-param>
    <param-name>session.secure</param-name>
    <param-value>1</param-value>
  </init-param>
    ...
</filter>
```

L'attribut `Secure` doit être présent dans le cookie Cerbère pour toute application consultée uniquement en TLS.

2.3.11 - Paramètre : interdire l'accès au jeton de session Cerbère par le code Javascript

Ce paramètre est **facultatif**.

Sauf cas très particulier, ce paramètre ne doit pas être renseigné dans le filtre. Sa valeur par défaut convient.

Il est préférable de rendre inaccessible le jeton de session Cerbère au code Javascript d'une l'application. Ceci pour de diminuer le risque de vol de cookie par des vulnérabilités de type XSS notamment. Cette interdiction d'accès se fait par l'ajout de l'attribut "`HttpOnly`" au cookie Cerbère.
Cet attribut est par défaut ajouté au cookie Cerbère et n'a pas à lui être supprimé, sauf dans de rares situations.
Les requêtes Ajax classiques transmettent le cookie Cerbère, l'attribut `HttpOnly` ne doit pas être supprimé pour les application de typeAjax.

Syntaxe :
Nom du paramètre : `session.httponly`
valeur : `oui`|`1` pour forcer l'ajout de l'attribut `HttpOnly` au cookie Cerbère.
défaut : `oui`|`1`

Exemple :

```
<!-- Définition du filtre Cerbère. -->
<filter>
  <filter-name>FiltreCerbere</filter-name>
  <filter-class>i2.[...].FiltreCerbere</filter-class>
    <!-- Paramètres d'initialisation. -->
    ...
    <!-- Supprimer l'attribut HttpOnly du cookie Cerbère -->
  <init-param>
    <param-name>session.httponly</param-name>
    <param-value>0</param-value>
  </init-param>
    ...
</filter>
```

2.4 - Keystore des certificats de test

2.4.1 - Présentation

Il est possible de simuler une authentification par certificat dans la version de Cerbère bouchon en utilisant une authentification classique par mot de passe. Si le keystore des certicats de test contient un certificat associé au mel de l'utilisateur, alors ce certificat sera retourné par la méthode Utilisateur.getCertificat() comme si l'authentification avait au lieu par certificat.
La récupération du certificat et son analyse ne sont nécessaires que si le profil des certificats contient des informations utiles à l'application autre que celles fournies par les APIs Cerbère sur l'utilisateur.

Pour cela il est nécessaire de créer un keystore Java de type JKS contenant les certificats de test. Ceux-ci peuvent avoir un profil quelconque pourvu qu'il convienne aux besoin de l'application et qu'il contiennent le mail utilisateur dans l'attribut `CN` ou `sujectAltName`.

2.4.2 - Installation du keystore

Créer un keystore de type JKS. Ce keystore doit contenir :

- Le certificat de l'autorité ayant signé les certificats utilisateurs.
- Les certificats utilisateurs relié aux comptes du fichier cerbere-bouchon par leurs adresse mail (dans `CN` ou `sujectAltName`).
- Paraméter ce keystore dans le filtre Cerbère-bouchon par les paramètres :
  - `bouchon.cert.ks` : chemin indiquant l'emplacement du keystore. Si le keystore n'est pas trouvé, ce paramètre est ignoré (défaut : cerbere-bouchon-certificats.jks sous `WEB-INF/classes/`.)
  - `bouchon.cert.ks.passe` : mot de passe du keystore contenant les certificats de test (défaut : cerbere)

L'ensemble de ces certificats peut être importé dans le keystore par la commande `keytool` du JDK.

Un exemple de keystore et de certificats associés est fourni sous docs/certificats. Pour le tester avec les utilisateur de l'application exemple, il suffit de la placer sous WEB-INF/classes.

Exemple :

```
<filter>
    <filter-name>FiltreCerbere</filter-name>
       <filter-class>i2.application.cerbere.filtre.FiltreCerbere</filter-
class>
    <!-- paramètres d'initialisation -->
    ...
    <!-- certificats de test -->
    <init-param>
        <param-name>bouchon.cert.ks</param-name>
```

```
        <param-value>/home/dev/cert-dev.jks</param-value>
    </init-param>
    <init-param>
        <param-name>bouchon.cert.ks.passe</param-name>
        <param-value>secret</param-value>
    </init-param>
    ...
</filter>
```

3 - Fichier référentiel Cerbère-bouchon XML

Cerbère-bouchon obtient les informations sur les utilisateurs, les applications, les entités et les droits par lecture d'un fichier XML, défini par le paramètre conf du filtre (`WEB-INF/cerbere-filtre-bouchon.xml` par défaut).
Le fichier XML est décrit et validé par un schéma XML, fourni dans la distribution (`/docs/annuaire.xsd`).

3.1 - Exemples

La distribution Cerbère contient un exemple de référentiels XML sous `docs/` : `cerbere-filtre-bouchon.xml`.

3.2 - Validation du fichier XML

Le fichier XML est validé par une schéma XML, fourni dans la distribution (`/docs/annuaire.xsd`) et présent dans la librairie Cerbère-bouchon. Si le fichier XML est incorrect, le filtre ne démarrera pas et lèvera une exception contenant l'erreur rencontrée dans le fichier XML.

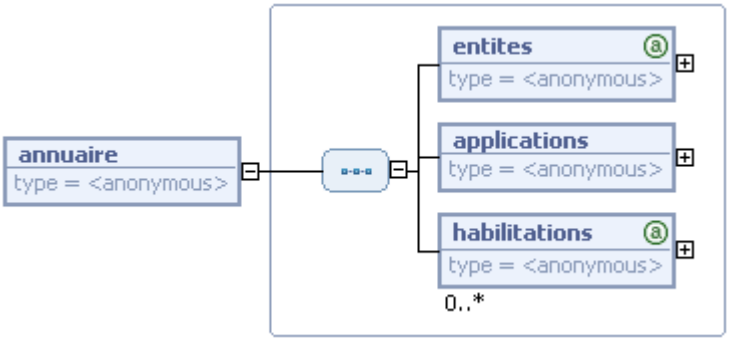
En complément et pour simplifier la validation du fichier XML, un script de validation est fourni dans la distribution (bin/XMLValide(.bat|.sh)). Sa syntaxe est la suivante :

- `XMLValide.bat -f <`*`fichierXML`*`>`sous Windows
- `XMLVallide.sh -f <`*`fichierXML`*`>` sous Linux

Avec
<*fichierXML*> : chemin du fichier XML à valider

3.3 - Conception du fichier référentiel XML

Le fichier référentiel XML est formée de 3 parties principales : les entités/utilisateurs, les applications et les habilitations.



3.3.1 - Éléments génériques

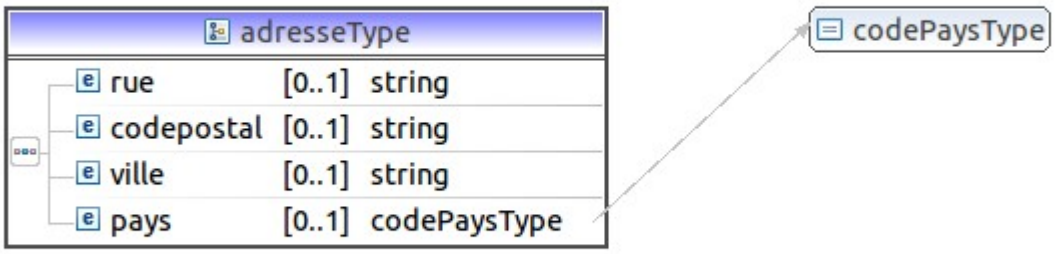
3.3.1.a - Booléen

Les attributs ou éléments de type booléen (oui/non) doivent prendre l'une des valeurs "oui", "non", "1" ou "0".

3.3.1.b - Adresse

Une adresse de personne ou d'entreprise est constituée :

- d'un nom de rue.
- d'un code postal.
- d'une ville.
- d'un code pays sur 2 chiffres. Ce code pays est déterminé selon la norme ISO-3166/A2 (FR pour France par exemple). La liste des codes pays est fournie dans la documentation.

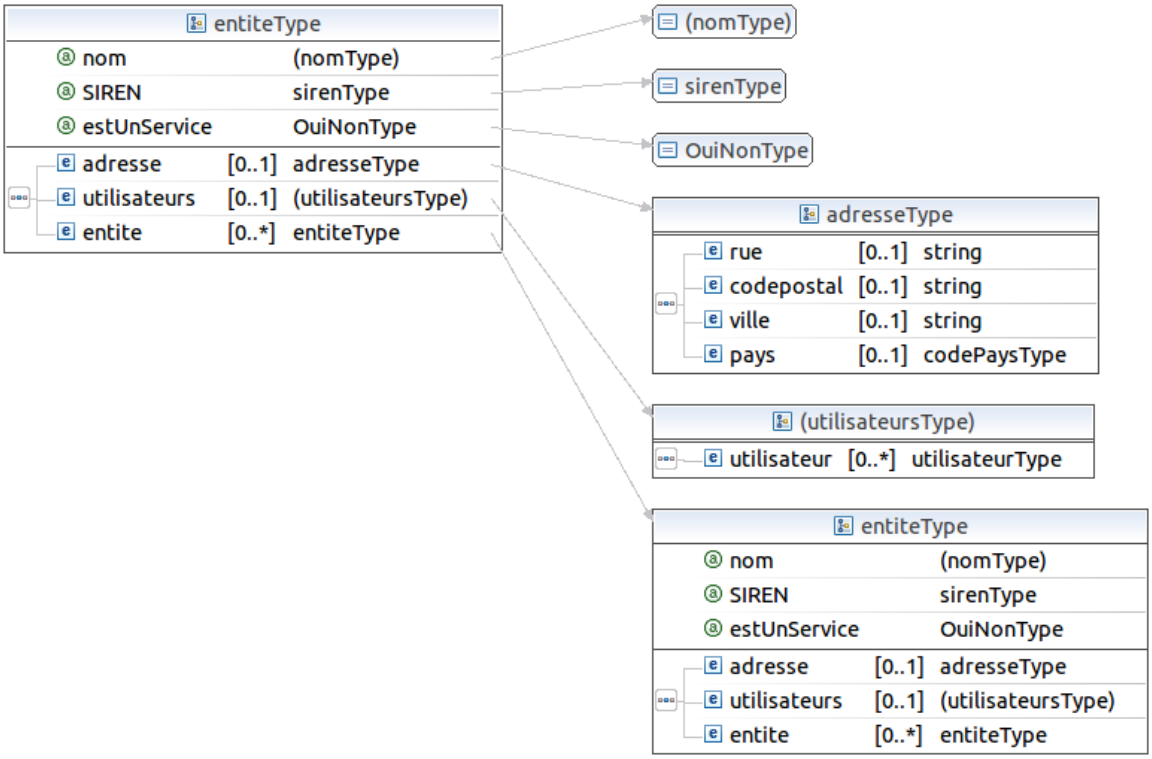


3.3.2 - Définition des entités

3.3.2.a - Définition générale.

Une entité est définie dans un élément `<entite>`. Un élément `<entite>` est formé (**dans l'ordre**) :

- d'un attribut `nom` (**obligatoire**) : nom de l'entité ou raison sociale pour une entreprise.
- d'un attribut `SIREN` (facultatif) : Numéro SIREN de l'entreprise.
- d'un attribut `estUnService` (facultatif) : indique si cette entité est ou non un service, non par défaut.
- d'un sous-élement `<adresse>` (facultatif) : adresse de l'entreprise.
- d'un sous-élement `<utilisateurs>` (facultatif) : définition des utilisateurs de l'entité, voir ci-après.
- de sous-élements `<entite>` (facultatif) : sous-entités de l'entité courante. La définition est récursive.



Remarques : le nom d'entité ne doit pas comporter de délimiteur de niveau (/). Pour créer par exemple l'entité SG/SPSSI/PSI, créer l'entité SG, puis la sous-entité SPSSI et à nouveau la sous-entité PSI. L'exemple cerbere-filtre-bouchon.xml illustre cela.

3.3.2.b - Cas d'une entité de type service.

Les entités de niveau service (Direction d'administration centrale, DDT, DREAL, entreprise ou Particuliers) doivent être marqués comme tels dans le fichier par l'attribut `estUnService` à '1" (ou "oui").

Un service ne peut contenir un autre service à un niveau inférieur (règle de gestion de l'annuaire).

Ce marquage est nécessaire est permet la calcul de bonnes valeurs pour l'attribut "Unité" des personnes.

Exemple : Pour la création de l'entité SG/SPSSI/PSI/PSI4 il faut créer :

- Une entité SG avec estUnService="1".
- Des sous entités SPSSI, PSI et PSI4 sans attribut estUnservice.

3.3.2.c - Cas d'une entreprise.

Une entreprise doit être marquée comme service (`estUnService="1"`) et doit contenir un attribut SIREN. Ce numéro SIREN sera validé syntaxiquement.

3.3.2.d - Gestion des adresses.

Les adresses sur les entités ne sont autorisées que pour les entreprises (les adresses sur les personnes sont autorisées à tout niveau).

3.3.3 - Définition des utilisateurs.

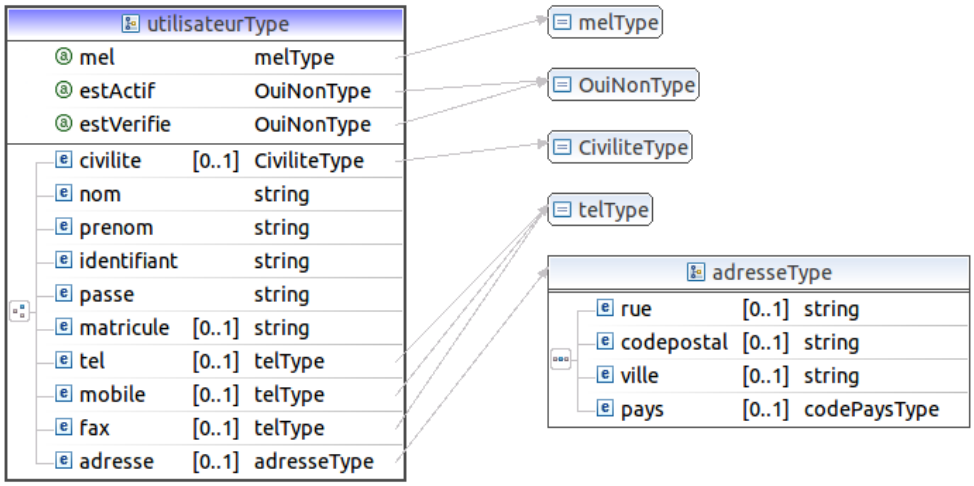
3.3.3.a - Définition générale.

Les utilisateurs sont définis au sein de leurs entités d'appartenance, dans l'élément `<utilisateurs>`. Cet élément contient autant de sous-éléments `<utilisateur>` que de nécessaire. Un utilisateur est défini par :

- un attribut `mel` (**obligatoire**) : adresse électronique de l'utilisateur;
- un attribut `estActif` (facultatif) : oui si utilisateur actif (défaut), non si désactivé;
- un attribut `estVerifie` (facultatif) : oui si l'utilisateur est vérifié (défaut), non sinon.
- un sous-élement `<civilité>` : civilité de l'utilisateur (M|F).
- un sous-élement `<nom>` (**obligatoire**) : nom de l'utilisateur dans le système Cerbère-bouchon;
- un sous-élement `<prénom>` (**obligatoire**) : prénom de l'utilisateur dans le système Cerbère-bouchon;
- un sous-élement `<identifiant>` (**obligatoire**) : identifiant d'accès au système Cerbère-bouchon local;
- un sous-élément `<passe>` (**obligatoire**) : passe d'accès au système Cerbère-bouchon local;
- un sous-élément `<matricule>` (facultatif) : matricule RH;
- un sous-élément `<tel>` (facultatif) : numéro de téléphone fixe;
- un sous-élément `<mobile>` (facultatif) : numéro de téléphone mobile;
- un sous-élément `<adresse>` (facultatif) : adresse postal;

► Les adresses électroniques et identifiant doivent être uniques.

► Il doit exister une entité de type service (estUnService=1) dans l'arborescence de l'utilisateur.



3.3.3.b - Niveau de confiance des comptes de particuliers et de professionnels.

Les comptes de particuliers et de professionnels sont par défaut marqués dans Cerbère comme non vérifiés. Pour reproduire cela dans la version bouchon ces comptes doivent posséder l'attribut `estVerfié="0"`.

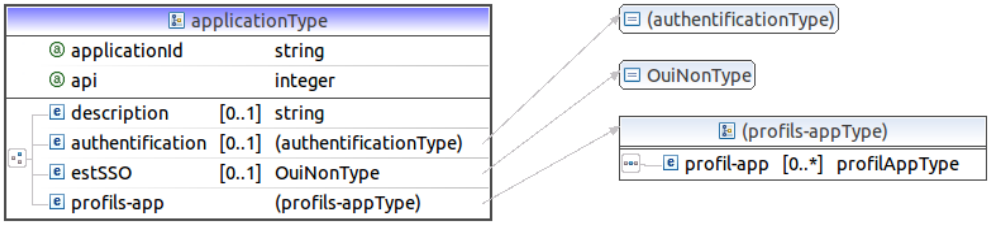
Les comptes créés dans Cerbère-bouchon sont par défaut considérés comme vérifiés.

3.3.4 - Définition des applications.

3.3.4.a - Les applications

L'élément `<applications>` contient un sous-élément `<application>` par application. L'élément `<application>` contient :

- un attribut `applicationId` (**obligatoire**) : identifiant de l'application, tel que renseigné dans le filtre.
- un attribut `api` (**obligatoire**) : indique s'il l'utilisation des API de recherche est autorisée (autorisé par défaut).
- un sous-élément `<description>` (facultatif) : description libre de l'application.
- Un sous-élément `<authentification>` (facultatif) : niveau d'authentification requis par l'application. Au choix `FORM` (défaut) pour formulaire ou `CERT` pour certificat (non implémenté pour l'instant).
- un sous-élément `<estSSO>` (facultatif) : accès autorisé ou non en SSO (Single Sign On). Au choix `oui` (défaut) ou `non`.
- Un sous-élément `<profils-app>` (**obligatoire**) : liste les profils d'application (voir ci-après).



3.3.4.b - Les profils d'application

Un profil d'application est défini par un élément `<profil-app>` sous l'élément `<profils-app>` de l'application. Il contient :

- un attribut `nom` (**obligatoire**) : identifiant du profil d'application.
- un sous-élément `<description>` (facultatif) : description libre du profil d'application.

3.3.5 - Définition des habilitations

Les habilitations sont regroupées dans le fichier XML par application (élément `<habilitations>`) puis par utilisateur (élément `<habilitation>`).

Un élément `<habilitations>` regroupe toutes les habilitations d'une même application et contient :

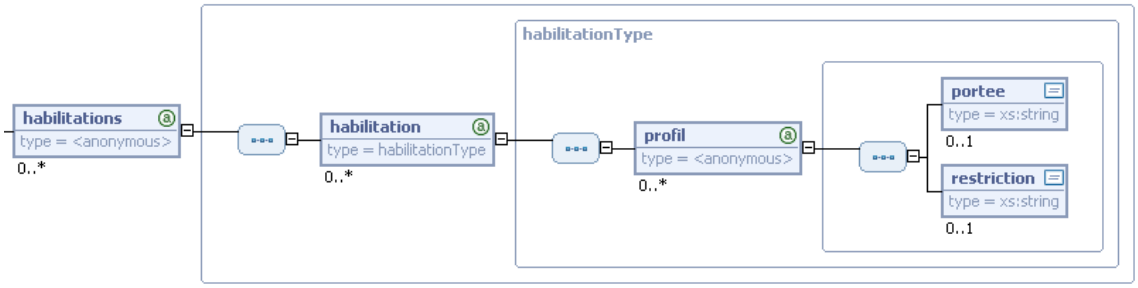
- un attribut `applicationId` (**obligatoire**) : identifiant de l'application.
- des sous-éléments `<habilitation>` (facultatif) : droits utilisateurs sur l'application.

Un élément `<habilitation>` regroupe les profils d'un utilisateur sur une même application et contient :

- un attribut `mel` (**obligatoire**) : adresse électronique de l'utilisateur (doit avoir été précédemment décrit dans noeud `<utilisateur>`).
- des sous-éléments `<profil>` (facultatif) : profils de l'utilisateur sur l'application.

Un élément `<profil>` défini un profil utilisateur sur l'application et contient :

- un attribut `nom` (**obligatoire**) : identifiant du profil (doit avoir été précédemment décrit dans noeud `<profil-app>`).
- un sous-élément `<portee>` (facultatif) : portée de service du profil.
- un sous-élément `<restriction>` (facultatif) : restriction complémentaire au profil.



4 - Test des API Cerbère avant mise en ligne

Un script en ligne de commande est fourni pour tester les API Cerbère sur le fichier XML sans avoir à déployer une application web. Ce script, présent sous `bin/CerbereTest`(`.bat`|`.sh`) a pour syntaxes :

● `CerbereTest -f <fichier> -a <applicationId> -u <login> -p <passe>`
avec
<*fichier*> : chemin du fichier référentiel XL
<*applicationId*> : identifiant application, tel que indiqué en initialisation du filtre
<*login*> : identifiant utilisateur
<*passe*> : passe utilisateur
► indique si l'accès est autorisé ou non à l'utilisateur (si refus, message d'erreur identique à celui du formulaire d'authentification)

Exemples :

```
./CerbereTest.sh -f docs/cerbere-filtre-bouchon.xml -a cerbere-test -u alain.martin-sg -w bouchon
=> utilisateur OK

./CerbereTest.sh -f docs/cerbere-filtre-bouchon.xml -a cerbere-test -u alain.martin-sg -w mauvais
=> mauvais mot de passe
```

● `CerbereTest -f <fichier> -a <applicationId> -u <login> -h`
avec
<*fichier*> : chemin du fichier référentiel XL
<*applicationId*> : identifiant application, tel que indiqué en initialisation du filtre
<*login*> : identifiant utilisateur
► liste les objets `Utilisateur`, `Application`, `Entreprise` et `Profil` des API Cerbère.

■