SG/SPSSI/CPII
DOSO/PNE Sécurité

09/05/2017

RAPPORT

Portail Cerbère d’authentification et d'autorisations

Interface de programmation Cerbère pour les applications.

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
09/05/2017	Ajout de l'attribut login
28/02/2017	Ajout de l'attribut Bureau. Précisions et corrections sur les APIs de recherche.
26/08/2015	Ajout des méthodes de déconnexion avec url de retour et de création de l'objet Cerbère sans argument.
28/08/2013	Remise à la charte.
03/05/2012	Intégration des entreprises et professionnels.
04/08/2011	Remise en forme.
29/11/2009	Version initiale.

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 et de l'interface de programmation

La librairie Cerbère contient le filtre d'authentification et l'ensemble des APIs Cerbère qui permettent l'authentification et l'habilitation des utilisateurs dans les applications ACAI au travers du portail de sécurité Cerbère.

L'interface de programmation (API) Cerbère permet aux applications du ministère du développement-durable d'interagir avec Cerbère 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 (cas des comptes de professionnels);

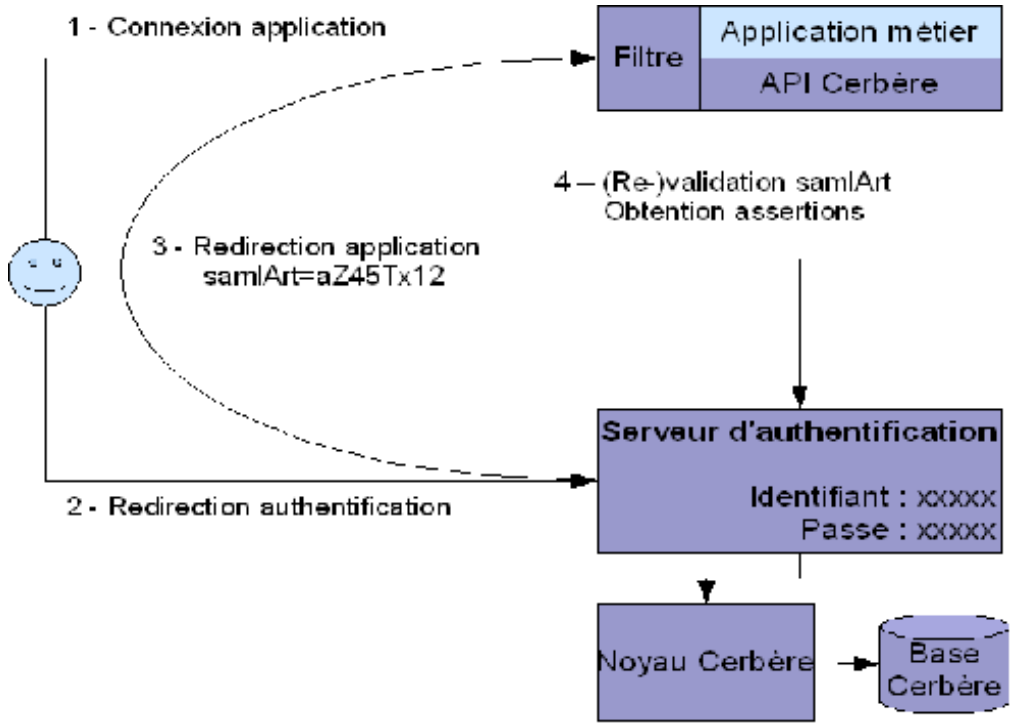
- d'obtenir des informations sur les droits de cet utilisateur (profils applicatifs);
- d'agir sur la connexion de l'utilisateur (ré-authentification ou déconnexion).

Ce document décrit cette interface de programmation.

1.2 - Fonctionnement général du filtre Cerbère.

Le filtre Cerbère garantit que tout utilisateur vu dans une application a été au préalable authentifié sur Cerbère.
Il intercepte les requêtes sur l'application et ramène l'utilisateur vers le portail d'authentification Cerbère s'il n'est pas authentifié ou si ses autorisations sont périmées.

Son fonctionnement général est celui de tout portail d'authentification, schématisé comme suit dans le cas d'un premier accès à une application métier sécurisée par Cerbère :



Les étapes suivies sont les suivantes :

1. L'utilisateur accède à son application métier par l'adresse (URL) de cette application.
2. Le filtre Cerbère ne reconnaît pas l'utilisateur et le redirige vers le portail d'authentification.

3. L'utilisateur s'authentifie (mot de passe ou certificat selon les exigences de l'application), le portail Cerbère le renvoi ensuite vers l'application avec un ticket d'authentification.
4. Le filtre Cerbère vérifie le ticket d'authentification sur le portail Cerbère et obtient en retour les informations d'authentification et d'autorisation sur cet utilisateur.
5. L'utilisateur est authentifié sur l'application (un jeton de session secondaire est déposé dans son navigateur sous forme d'un cookie de session) et le filtre Cerbère laisse passer la requête vers l'application métier.
6. Le jeton secondaire est vérifié lors de chaque requête ultérieure à l'application métier. Si ce jeton est absent, invalide ou périmé l'utilisateur est à nouveau redirigé vers le portail d'authentification Cerbère.

2 - Organisation de l'annuaire Cerbère.

L'annuaire Cerbère contient des personnes et des entités. Il reflète l'annuaire Amande du ministère du Développement-durable pour les comptes présents dans Amande.

2.1 - Les différents types de compte Cerbère.

Les comptes Cerbère peuvent provenir de l'annuaire Amande du ministère, être créés manuellement par des administrateurs Cerbère ou être créés directement par le propriétaire du compte (particuliers et professionnels).

2.2 - Les comptes issus de l'annuaire Amande du ministère.

Les **agents du ministère du Développement durable (DDT comprises)** disposant d'une boite aux lettre individuelle dans la messagerie Melanie2 du ministère se voient automatiquement attribuer un compte Cerbère.
Ces compte sont positionnés dans Cerbère à un emplacement similaire à celui de l'annuaire Amande.

Les **agents des directions départementales interministérielles (DDI, hors DDT)** sont présents dans l'annuaire Amande, ils disposent d'un compte Cerbère.
Ces comptes sont positionnés dans Cerbère à un emplacement similaire à celui de l'annuaire Amande, c'est à dire dans un service "Fimad". Les services "Fimad" sont nommés "<*Numéro département*> <*NOM DEPARTEMENT*>" (convention du ministère de l'Intérieur), par exemple :

- 01 AIN
- 04 ALPES-DE-HAUTE-PROVENCE
- 22 COTES-DARMOR

Les **agents des services déconcentrés du Ministère de l'Agriculture (DRAFF, DAF, DAAF)** sont présents dans l'annuaire Amande, ils disposent d'un compte Cerbère.
Ces comptes sont positionnés dans Cerbère à un emplacement similaire à celui de l'annuaire Amande (DRAAF-XXX, DAAFNN, DAFNNN).

2.2.1 - Les comptes de particuliers.

Toute personne peut créer son propre compte sur le portail Cerbère en tant que particulier.

Les comptes de particuliers sont tous regroupés au même niveau dans un service virtuel appelé "Particuliers".

2.2.2 - Les comptes de professionnels.

Tout professionnel (privé ou public) peut créer sont propre compte sur le portail Cerbère. Ce compte doit être associé à une entreprise.
Une entreprise est définie par son numéro SIREN et sa raison sociale (nom). La raison sociale retenue est celle fournie par le répertoire SIRENE de l'INSEE.

Les comptes de professionnels sont regroupés dans des services virtuels nommés selon le numéro SIREN : "SIREN <*Numéro SIREN*>".
Exemple : "SIREN 552049447" pour la SNCF.

2.3 - L'organisation par service.

2.3.1 - La notion de service.

Les comptes Cerbère sont classés dans des entités organisées autour de services. Un service est par exemple une entité correspondant à une entreprise ou à un nœud de l'annuaire Amande marqué comme tel.

L'une des particularité d'un service est l'unicité de son nom dans l'annuaire.

● Les services de l'annuaire Amande correspondent aux services administratifs du ministère. Ce sont par exemple :

- En administration centrale : SG, DGITM, DGPR, CGDD, …
- En région (DREAL) : DREAL Alsace, DREAL Aquitaine, ...
- En département (DDTM /DDTM) : DDT 01, DDT 02, …, DDTM 06, …
- Les services techniques centraux ou déconcentrés : CETE Xxx, SERTA, CERTU, ...
- Les services de la navigation (SN Xxx).
- Les directions interdépartementales des routes (DIRXXX).
- Les direction inter-régionales de la mer (DIRM XXX).
- Le CGEDD et les services d'inspection (IGAM, IGEM, ..).

Note : Tous ces services peuvent obtenus sur l'annuaire Amande par la requête LDAP suivante : "(mineqTypeEntree=NSER)".

● Les comptes de particulier sont tous regroupés dans un service virtuel nommé "Particuliers".

● Les comptes de professionnel sont regroupés par entreprise dans des services nommés "SIREN Xxx" (Xxx est le numéro SIREN de l'entreprise).

2.3.2 - La notion d'unité.

Un service peut contenir des entités filles (ou sous-niveaux d'organisation).

L'unité d'une personne décrit son emplacement dans l'annuaire depuis son service jusqu'à l'entité de plus bas niveau dans laquelle se trouve cette personne.

Exemples :

- "SG/SPSSI/PSI/PSI4" pour un agent du bureau PSI4.
- "DREAL Aquitaine/SCE/Division Energie" pour un agent du service Énergie de la DREAL Aquitaine.
- "Particuliers" pour tout compte de particulier.
- "SIREN 552049447" pour un employé de la SNCF.

2.4 - La vérification des comptes ou leurs niveaux de confiance.

2.4.1 - Le niveau de confiance d'un compte.

Toute personne peut se créer un compte Cerbère avec des informations libres et non forcément valides (nom, prénom, …). Cerbère ne vérifie que le courriel de façon systématique. Un compte de particulier ou de professionnel ne reflète à priori pas forcément l'identité réelle de la personne.

Afin de permettre aux applications de gérer cette incertitude sur ces comptes, u**n niveau de confiance (ou vérification) est associé à chaque compte**. Un compte vérifié contient de données qui ont été validées.

Les comptes issus de l'annuaire Amande sont automatiquement vérifiés.
Un compte externe crée manuellement par un administrateur Cerbère est considéré comme vérifié.

Un compte de particulier ou de professionnel est par défaut non vérifié.

2.4.2 - La vérification d'un compte ou l'augmentation de son niveau de confiance.

La vérification d'un compte de particulier ou de professionnel est une **action manuelle faite par un administrateur Cerbère**. Cette vérification ne peut se faire que dans le cadre d'un échange externe à Cerbère entre la personne et l'administration, généralement dans le cadre des procédures de l'application métier qui utilise Cerbère.

Il est possible pour un professionnel de disposer d'emblée d'un compte vérifié. Il lui faut pour cela créer son compte avec un certificat numérique reconnu par Cerbère et attestant de son identité.
Un compte de professionnel crée avec un certificat numérique est automatiquement vérifié.

2.4.3 - La prise en compte par les applications du niveau de confiance.

Chaque application est configurée dans Cerbère pour accepter ou non les comptes non vérifiés. Cette configuration est soit globale à l'application soit par profil.

Par défaut les applications sont configurées dans Cerbère pour n'accepter que les comptes vérifiés.

Dans la cas où une application accepte les comptes non vérifiés, une méthode des APIs Cerbère lui permet de de connaître le niveau de confiance du compte pour lui appliquer les traitements adéquats.

3 - Utilisation de l'interface de programmation.

3.1 - Objets Cerbère.

L'accès aux informations retournées par Cerbère se fait par une interface de programmation java (API) située dans le paquetage `i2.application.cerbere`.

Attention :
L'utilisation de classes Cerbère non documentées, en dehors du paquetage `i2.application.cerbere`, est interdite dans les applications (notamment celles issues des paquetages internes `cerberemoe`.* et `cerbere.*`). Seules les classes documentées ci-dessous font parties de l'API Cerbère. Toute classe technique ou interne présente dans les librairies Cerbère et non documentée dans ce document est susceptible de changer à tout moment.

La classe **`i2.application.cerbere.commun.Cerbere`** est le point d'entrée de l'API, elle dispose d'une méthode statique **`creation`** qui permet de créer une instance de la classe `Cerbere`. Cette méthode renvoie une exception `CerbereConnexionException` si un problème est intervenu pendant la connexion au serveur.

L'accès aux objets Cerbère se fait comme suit :

```
Cerbere cerbere = Cerbere.creation();
```

***Note** : la méthode creation(requeteHTTP) est identique à la méthode sans argument. Elle reste présente pour compatibilité ascendante mais n'est plus utile.*

À partir de l'objet `Cerbere`, il est possible d'accéder aux différentes informations concernant l'utilisateur :

- Sa carte d'identité : `i2.application.cerbere.commun.Utilisateur`.
- Son entreprise (professionnels uniquement) : `i2.application.cerbere.commun.Entreprise`.
- Ses habilitations : `i2.application.cerbere.commun.Habilitation`.
- Ses informations sur l'application : `i2.application.cerbere.commun.Application`.

La classe `Cerbere` délivre ces objets par les méthodes `getUtilisateur()`, `getEntreprise()`, `getHabilitation()` et `getApplication()` respectivement.

+ Classes :

```
i2.application.cerbere.commun.Cerbere
```

+ Constantes :

```
public static final int FORMULAIRE
public static final int CERTIFICAT
public static final int CERTIFICAT_1
public static final int CERTIFICAT_2
public static final int CERTIFICAT_3
```

+ Méthodes :

```
public static Cerbere creation();
public static Cerbere creation(requeteHTTP);

public Utilisateur getUtilisateur();
public Habilitation getHabilitation();
public Application getApplication();
public Entreprise getEntreprise();

public void reAuthentification(requeteHTTP, reponseHTTP, urlRetour);
```

```
public void reAuthentification(requeteHTTP, reponseHTTP, urlRetour,
niveau);
public void logoff(requeteHTTP, reponseHTTP);
public void logoff(requeteHTTP, reponseHTTP, urlRetour);

public Entite findEntite(entite);
public Utilisateur findUtilisateurByMel(mel);
public List findUtilisateurs(...);
public List findUtilisateursMel(...);
public List findUtilisateursAttributs(...);
```

3.2 - Informations sur l'utilisateur (*getUtilisateur*).

L'objet `Personne` contient les informations d'identité d'un compte Cerbère, à l'exception des paramètres de sécurité. L'objet `Utilisateur` complète l'objet `Personne` par les informations de sécurité, il hérite de l'objet `Personne`.
Une instance de l'objet `Personne` peut être obtenue soit par sa classe dérivée `Utilisateur` (`Cerbere.getUtilisateur`), soit par les méthodes de recherche (`Entite.findPersonnes`).

3.3 - Propriétés d'identité.

Les propriétés de l'utilisateur peuvent être obtenues par un accesseur dédié ou par la méthode générique `getValeur(nomPropriété)` :

Description	Méthode	Nom de la propriété (Personne.XXX)
Civilité de la personne. Les valeurs possibles sont 'M' pour un utilisateur masculin, 'F' pour une utilisatrice ou *null* si non présent dans Cerbère.	getCivilite()	CIVILITE
Nom de la personne.	getNom()	NOM
Prénom de la personne.	getPrenom()	PRENOM
Matricule de ressources humaine ("*code RH*") de la personne, *null* si non présent dans Cerbère.	getMatricule()	MATRICULE
Adresse électronique.	getMel()	MEL
Numéro de **téléphone fixe** de la personne, *null* si non présent dans Cerbère.	getNumTelFixe()	TEL
Numéro de **téléphone mobile** de la personne, *null* si non présent dans Cerbère.	getNumTelMobile()	TEL_MOBILE
Numéro de **fax** de la personne, *null* si non présent dans Cerbère.	getNumFax()	FAX
Composante "**rue**" de l'adresse postale de la personne, *null* si non présente dans Cerbère.	getAdresseRue()	ADR_RUE
Composante "**code postal**" de l'adresse postale de la personne, *null* si non présente dans Cerbère.	getAdresseCodePostal()	ADR_CODEPOSTAL
Composante "**ville**" de l'adresse postale de la personne, *null* si non présente dans Cerbère.	getAdresseVille()	ADR_VILLE
Code pays 2 lettres (ISO 3166-1 A2) du pays de la personne, *null* si non présent dans Cerbère.	getAdressePaysCode()	ADR_PAYS_CODE
Nom du pays de la personne, *null* si non présent dans Cerbère.	getAdressePaysNom()	ADR_PAYS_NOM

Unité complète de la personne ("*Service/Unite/Sous-Unite/...*")	getUnite()	UNITE
Identifiant interne Cerbère de la personne.	getIdentifiant()	ID
Bureau de la personne	getBureau()	BUREAU
Login (identifiant de connexion)	getLogin()	LOGIN

→ Exemple de lecture du nom de la personne :
- `objetUtilisateur.getNom()`
- `objetUtilisateur.getValeur(Personne.NOM)`

3.3.1 - Vérification du compte ou niveau de confiance.

La méthode `isVerifie()` permet de savoir si les données du compte (nom, prénom, …) ont été vérifiés.

→ Méthode :
```
public boolean isVerifie()
```

Cette méthode renvoie `true` si les données du compte sont considérées comme valides, `false` sinon. Par défaut les comptes de particuliers et de professionnels créés sans certificat numérique sont non vérifiées (`false`), les autres comptes sont vérifiées (`true`).

3.3.2 - Propriétés de sécurité.

Les propriétés de sécurité de la personne ne sont accessibles que par la classe `Utilisateur`.

→ Méthode :
```
public int getAuthNiveau()
```

Cette méthode renvoie le niveau d'authentification utilisé par l'utilisateur. Les niveaux d'authentifications disponibles sont (constantes de l'objet `Cerbere`) :
- `Cerbere.FORMULAIRE` : authentification par formulaire avec identifiant et mot de passe
- `Cerbere.CERTIFICAT_1` : authentification par certificat 1 étoile (idem `Cerbere.CERTIFICAT`).
- `Cerbere.CERTIFICAT_2` : authentification par certificat 2 étoiles.
- `Cerbere.CERTIFICAT_3` : authentification par certificat 3 étoiles.

→ Méthode :
```
public X509Certificate getCertificat()
```

Si l'utilisateur s'est authentifié par certificat, cette méthode retourne le certificat X509 employé. **Si l'utilisateur s'est identifié par mot de passe cette méthode retourne null.**

Dans le cas particulier d'une authentification par certificat, cette méthode donne accès au certificat X509 employé afin que l'application puisse l'exploiter.

3.4 - Informations sur l'application (*getApplication*).

Ces informations sont accessibles depuis la méthode `getApplication` de l'objet `Cerbère`.

Description	Méthode
Nom de l'application tel qu'affiché sur la page d'authentification.	getNom()
Vrai (*true*) si l'application accepte une authentification unique entre applications (**SSO**), faux (*false*) sinon.	isSSO()

Niveau d'authentification minimum exigé par l'application. Les niveaux d'authentifications disponibles sont identiques aux niveaux d'authentification de l'utilisateur.	getAuthNiveau()

3.5 - Autorisations de l'utilisateur sur l'application (*getHabilitation*).

Les autorisations de l'utilisateur se trouvent dans un objet de type `Habilitation` accessible par la méthode `getHabilitation` de l'objet `Cerbere`.
L'objet habilitation contient des méthodes utilitaires qui permettent de lire l'équivalent d'un tableau de droits de l'utilisateur sur l'application. Ce tableau de droits est constitué d'une liste de profil, chaque profil étant lui-même un triplet {nom, portée, restriction}.

3.5.1 - Description d'un profil.

Un profil utilisateur est composé de trois éléments.

- Le **nom** du profil, obligatoire, décrit la fonction du profil (ADMINISTRATEUR, GESTIONNAIRE par exemple). La définition des profils est à la charge du maître d'œuvre de l'application.
- La **portée** du profil, toujours présente, référence un nœud de l'annuaire Cerbère (et donc de l'annuaire Amande du ministère pour les comptes internes). Cette information permet à l'application d'appliquer des filtrage sur les traitements selon par exemple le service gestionnaire d'un dossier.
- La **restriction** complémentaire, facultative, est une information libre dont le format et l'interprétation sont à la seule charge de l'application.

Un utilisateur d'application peut disposer de plusieurs profils d'un même nom, ces profils se différenciant entre eux par la portée ou la restriction.

Description	Méthode
Nom du profil, toujours retourné en **majuscule**.	getNom()
Portée du profil sous la forme d'une unité.	getPortee()
Niveau d'authentification minimum exigé par l'application. Les niveaux d'authentifications disponibles sont identiques aux niveaux d'authentification de l'utilisateur.	getRestriction()

→ Méthode :
```
public String getNom()
```

Cette méthode permet de récupérer le nom du profil.
Le nom du profil est toujours retourné en majuscules.

→ Méthode :
```
public String getPortee()
```

Cette méthode permet de récupérer la portée associée au profil de l'application.

→ Méthode :
```
public String getRestriction ()
```

Cette méthode permet de récupérer la restriction complémentaire associée au profil de l'application de l'objet `Profil`. La méthode retourne `null` si aucune restriction complémentaire n'est définie sur ce profil.

3.5.2 - Lister les noms des profils.

```
public String[] getNomsProfils()
```

Cette méthode retourne un tableau contenant les noms des profils de cet utilisateur.

3.5.3 - Lister tous les profils.

```
public List<Profil> getTousProfils()
```

Cette méthode retourne la liste de tous les profils utilisateur dans l'application courante.

Important : Cette méthode remplace la méthode dépréciée `ArrayList getAllProfils();`

3.5.4 - Rechercher un profil par son nom.

```
public List<Profil> getProfilsParNom(String nomProfil)
```

Cette méthode retourne la liste des profils utilisateur d'un nom donné. Cette méthode retourne `null` si cet utilisateur ne possède aucun profil de ce nom.
Le paramètre `nomProfil` n'est pas sensible à la casse.

Important : Cette méthode remplace la méthode dépréciée `ArrayList getProfils(String nomProfil);`

3.5.5 - Rechercher un profil par comparaison.

```
public boolean hasProfil(String nomProfil,String portee,String restriction)
```

Cette méthode permet de déterminer si l'utilisateur possède le profil d'application avec la portée et la restriction correspondante. La méthode retourne `true` si toutes les conditions sont vérifiées `false` sinon. La portée ou la restriction passées en paramètres peuvent être `null`, la méthode n'en tiendra alors pas compte.
Le paramètre `nomProfil` n'est pas sensible à la casse.

3.6 - Informations sur l'entreprise d'un professionnel (*getEntreprise*).

Si l'utilisateur courant correspond à un professionnel (enregistré dans une entreprise dans Cerbère), alors les APIs Cerbère donnent accès aux informations de cette entreprise. **Ces informations n'existent pas pour les autres types de compte utilisateur**.

Ces informations sont accessibles depuis la méthode `getEntreprise` de l'objet `Cerbère`.
Cette méthode retourne un objet de type `Entreprise` pour un compte de professionnel, `null` pour les autres comptes.

Description	Méthode
Raison sociale de l'entreprise.	getRaisonSociale()
SIREN de l'entreprise.	getSIREN()
Composante "**rue**" de l'adresse postale de l'entreprise, *null* si non présente dans Cerbère.	getAdresseRue()
Composante "**code postal**" de l'adresse postale de l'entreprise, *null* si non présente dans Cerbère.	getAdresseCodePostal()

Composante "**ville**" de l'adresse postale de l'entreprise, *null* si non présente dans Cerbère.	getAdresseVille()
Code pays 2 lettres (ISO 3166-1 A2) du pays de l'entreprise, *null* si non présent dans Cerbère.	getAdressePaysCode()
Nom du pays de l'entreprise, *null* si non présent dans Cerbère.	getAdressePaysNom()

3.6.1 - Ré-authentifier et déconnecter un utilisateur.

Il est possible que l'application souhaite la **ré-authentification** de l'utilisateur. Ce peut être par exemple sur un cas d'utilisation qui requière une authentification de niveau carte à puce alors que l'utilisateur s'est authentifié en mot de passe.
L'API Cerbère donne un moyen de déclencher cette procédure au travers de méthodes de la classe `Cerbere` :

```
public void reAuthentification(HttpServletRequest requeteHTTP,
        HttpServletResponse reponseHTTP,
        String urlRetour)
        throws CerbereConnexionException

public void reAuthentification(HttpServletRequest requeteHTTP,
        HttpServletResponse reponseHTTP,
        String urlRetour,
        int niveauAuthentification)
        throws CerbereConnexionException.java
```

Le paramètre `urlRetour` précise la page sur laquelle l'utilisateur reviendra une fois qu'il se sera ré-authentifié.
Les niveaux d'authentification disponibles sont identiques à ceux niveaux définis pour l'utilisateur.

Exemple :

```
// Création de l'objet Cerbere.
Cerbere cb = Cerbere.creation();

// Oblige l'utilisateur à se reconnecter par certificat,
// si la connexion réussit l'utilisateur est redirigé sur la page
valide.htm"

cb.reAuthentification(request, response, "valide.html",
Cerbere.CERTIFICAT_1);
```

Il est aussi possible de **déconnecter** de Cerbère un utilisateur. La méthode `logoff` effectue pour cela une redirection sur le portail Cerbère.

Classe : `i2.application.cerbere.commun.Cerbere`

→ Méthode :
`public void logoff(HttpServletRequest requeteHTTP,HttpServletResponse reponseHTTP).`

`public void logoff(HttpServletRequest requeteHTTP,HttpServletResponse reponseHTTP, urlRetour).`

Le paramètre `urlRetour` précise l'adresse vers laquelle sera redirigée l'utilisateur après sa déconnexion. En l'absence d'adresse, l'utilisateur restera sur le portail Cerbère.

3.7 - Méthodes de recherche.

Cerbère dispose de méthodes permettant de rechercher des personnes et des entités dans l'annuaire Cerbère.
L'application doit être autorisée à utiliser ces méthodes sans quoi une exception `CerbereException` sera levée.

L'utilisation de ces méthodes doit être bien réfléchie. Il est fortement conseillé de contacter le PNE Sécurité ou SG/SPSSI/PSI/PSI4 pour valider leurs emplois dès la phase de conception de l' application.

Les méthodes de recherche fonctionnement en mode connecté avec le serveur Cerbère, leurs utilisations ne sont pas transparentes pour l'application en terme de performance.

3.7.1 - Recherche d'une entité de l'annuaire.

Classe : `i2.application.cerbere.commun.Cerbere`

→ Méthode :
```
public Entite findEntite(String nom) throws CerbereException;
```

Cette méthode permet de rechercher une entité dans l'annuaire Cerbère. La valeur `null` est retournée si aucune entité de ce nom n'est trouvée. Une exception `CerbereException` est levée en cas de problème lors de la recherche.

Paramètres:
• `nom` : nom de l'entité à rechercher. Ce nom doit commencer par un service et peut contenir des sous entités.

Si la recherche porte sur une entreprise, le nom à utiliser est "SIREN <Numéro SIREN>".

Exemples de noms valides pour une recherche :
- "SG" : Rechercher dans le service SG '(Secrétariat général du ministère).
- "SG/SPSSI/PSI" : Limiter la recherche à la sous-direction PSI.
- ""SIREN 123456789" : Recherche dans l'entreprise dont le SIREN est 123456789.

L'objet `Entite` retourné contient deux attributs.

Description	Méthode
Nom court de l'entité ("SG", "PSI", "SIREN 12345679" par exemple).	getNom()
Nom complet (**UNITE**) de l'entité. *Exemple* : • SG • SG/SPSSI/PSI • SIREN 123456789	getUnite()

3.7.2 - Recherche de sous-entités d'une entité.

Classe : `i2.application.cerbere.commun.Entite`

→ Méthode :
```
public List findSousEntites() throws CerbereException;
```

Cette méthode permet de récupérer la liste des sous entités d'une entité donnée. Si aucune sous entité n'est trouvée, la liste est vide. Une exception `CerbereException` est levée en cas de problème lors de la recherche.

3.7.3 - Recherche des personnes d'une entité.

Classe : `i2.application.cerbere.commun.Entite`

Méthode :
→ `public List findPersonnes(boolean recursif) throws CerbereException;`

Cette méthode permet de récupérer la liste des personnes d'une entité donnée. Si l'entité ne contient aucune personne, la liste est vide. Une exception `CerbereException` est levée cas de problème lors de la recherche.
La liste contient des objets de type `Personne`.

Paramètres:
- `récursif` : `true` si recherche récursive dans tous les sous entité, `false` sinon.

3.7.4 - Recherche d'utilisateur de l'application par son courriel.

Classe : `i2.application.cerbere.commun.Cerbere`

→ Méthode :
```
public Utilisateur findUtilisateurByMel(String mel) throws
CerbereException;
```

Cette méthode permet de rechercher un utilisateur de l'application par son adresse électronique. Si aucune utilisateur n'est trouvé, `null` est retourné. Une exception `CerbereException` est levée en cas de problème lors de la recherche.

Paramètres:
- `mel` : adresse électronique de l'utilisateur recherché.

L'objet utilisateur retourné ne permet pas d'obtenir en cascade les autorisations de cet utilisateur. La méthode <utilisateur>`.getHabilitation()` renverra `null` sur un utilisateur résultant des APIs de recherche.

3.7.5 - Recherche multi-critères d'utilisateur de l'application.

Plusieurs méthodes permettent de rechercher des utilisateurs connus dans l'application. Les critères de recherche autorisés sont :
- `entite` : entité d'appartenance de l'utilisateur, ignorée si nul.
- `profils` : profil de l'utilisateur, ignoré si nul.
- `restriction` : restriction sur le profil, ignorée si nul.
  - Si ce paramètre est fourni, un profil doit également être fourni.
  - Ce paramètre peut être une expression régulière, selon la syntaxe comprise par la classe `java.util.regex.Pattern`. Il faut alors l'écrire "`/restriction/`" ou "`/restriction/i`", la deuxième forme correspond à une expression régulière insensible à la casse.

Les méthodes associées à ces critères sont :

→ Méthode :
```
public List  Utilisateur findUtilisateurs(String entite,  boolean
recursif,String profil, String restriction) throws CerbereException;
```

Recherche multi-critères permettant de récupérer une liste d'objets `Utilisateur` de l'application courante.

Paramètres:
- `entite` : cf. explications ci-dessus.
- `recursif` : `true` si recherche récursive parmi les entités, `false` sinon.

- `profil` : cf. explications ci-dessus.
- `restriction` : cf. explications ci-dessus.

L'objet utilisateur retourné par ces méthodes ne permet pas d'obtenir en cascade les autorisations d'un utilisateur. La méthode <utilisateur>`.getHabilitation()` renverra `null` sur un utilisateur résultant des APIs de recherche.

→ Méthode :
```
public List<String> findUtilisateursMel(String entite, boolean
recursif,String profil, String restriction) throws CerbereException;
```

Recherche multi-critères permettant de récupérer une liste d'adresses mel d'utilisateurs de l'application courante. Cette méthode est adaptée lorsqu'un grand nombre de résultats est attendu et que seul l'identifiant de l'utilisateur est nécessaire.

Paramètres:
- `entite` : cf. explications ci-dessus.
- `recursif` : `true` si recherche récursive parmi les entités, `false` sinon.
- `profil` : cf. explications ci-dessus.
- `restriction` : cf. explications ci-dessus.

→ Méthode :
```
public List<Map<String, String>> findUtilisateursAttributs(String entite,
            boolean recursif, String profil, String restriction,
            String attributs) throws CerbereException;
```

Recherche multi-critères permettant de récupérer une liste d'attributs utilisateurs de l'application courante déterminés par leurs seuls attributs nécessaires à l'action courante dans l'application. Cette méthode est adaptée lorsqu'un grand nombre de résultats est attendu et que seuls certains attributs utilisateurs sont nécessaires.

Paramètres:
- `entite` : cf. explications ci-dessus.
- `recursif` : `true` si recherche récursive parmi les entités, `false` sinon.
- `profil` : cf. explications ci-dessus.
- `restriction` : cf. explications ci-dessus.
- `attributs` : liste des attributs utilisateurs désirés, parmi ceux proposés par l'interface Personne.

3.8 - Dépréciations et incompatibilités avec les versions précédentes des APIs.

3.8.1 - Liste des incompatibilités.

La prise en compte des particuliers et des professionnels dans les API Cerbère est incompatible avec la notion d'entreprise telle que définie dans l'ancien système Cactus.

Les objets et méthodes des APIs Cerbère dédiés à Cactus ne fonctionnent plus en l'état dans les nouvelles APIs Cerbère.

La méthode **getEntite**() de l'objet Cerbère n'existe plus. Cette méthode n'était utile que dans le contexte Cactus et non utilisé dans le contexte Cerbère (elle renvoyait systématiquement *null*).
La méthode équivalente dans les nouvelles APIs est `getEntreprise()`.

L'objet **Entite** ne contient plus que les informations génériques d'une entité (nom et unité). Les informations propres à une entreprise sont regroupées dans l'objet Entreprise.

3.9 - Méthodes et attributs dépréciés.

Les méthodes suivantes ont été dépréciées. Elles restent valides mais doivent être remplacées à terme par des méthodes alternatives.

● **Profils de l'utilisateur**.

Les méthodes `getProfils` et `getAllProfils` de l'objet Habilitation sont remplacées respectivement par les méthodes `getProfilsParNom` et `getTousProfils` au fonctionnement équivalent mais plus clairement typées.

● **Attributs LDAP_O, LDAP_OU et LDAP_DN**.

Ces attributs sont dépréciés. Ils sont présents dans les objets Personne et Entite.

L'attribut UNITE permet de retrouver les informations présentes et pertinentes de ces attribut.
L'unité d'une personne représente son emplacement dans l'annuaire depuis son service jusqu'à l'entité de plus bas niveau dans laquelle se trouve la personne.

4 - Exemples.

4.1 - Exemple d'utilisation des classes et des méthodes de base.

```
try{
    // Création des objets Cerbere.
    Cerbere cerbere = Cerbere.creation();

    // nom de l'application
    Application application = cerbere.getApplication ();
    String appNom = application.getNom();
    //niveau minimum d'authentification requis par  l'application
    int appNiveauAuth = application.getAuthNiveau ();


    // informations de l'utilisateur (nom prénom et courriel)
    Utilisateur utilisateur = cerbere.getUtilisateur();
    String personneMel  = utilisateur.getMel() ;
    String personneNom  = utilisateur.getNom() ;
    String personnePrenom  = utilisateur.getPrenom() ;

    // niveau d'authentification de l'utilisateur courant
    int personneNiveauAuth = utilisateur.getAuthNiveau();

    // SIREN de l'entité d'appartenance de l'utilisateur
    Entreprise entreprise = cerbere.getEntreprise();
          String  etsSIREN  =  entreprise.getSIREN();  //  Comptes  de
professionnels seuls.

    // Récupérer les habilitations de l'utilisateur.
    Habilitation habilitation= cerbere.getHabilitation();
    List<Profil> profils = habilitation.getTousProfils();

    // test de la présence du profil "ADMINISTRATEUR"
    if (habilitation.hasProfil ("ADMINISTRATEUR", null, null) {
        // Traitements pour un administrateur.
    }
} catch (CerbereConnexionException e) {
    // Traitement "Impossible de récupérer l'objet Cerbere" ;
}
```

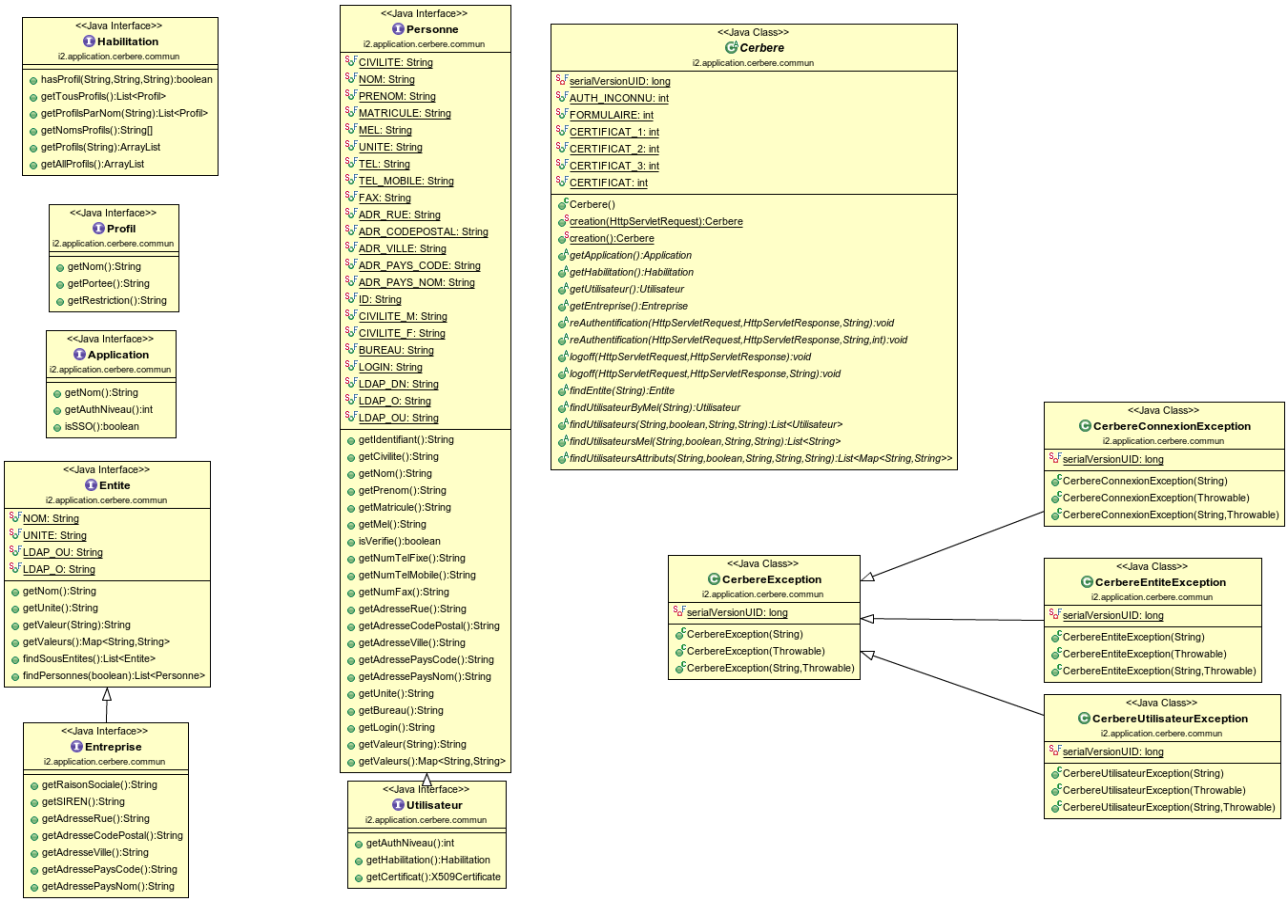
4.2 - Exemple d'utilisation des classes et de méthodes de recherche.

```
/*
Pré-requis : l'objet Cerbere est créé et disponible dans la variable
"cerbere".
*/

// recherche des tous les gestionnaires de l'application
try{
  List<Utilisateur> gestionnaires = cerbere.findUtilisateurs(
      null, false, "GESTIONNAIRE", null);
  // parcours de ces gestionnaires
  for( Utilisateur gestionnaire :  gestionnaires) {
    // ... suite du traitement
  }
```

```
} catch (CerbereException e) {
    // Traitement en cas d'exception.
}

// recherche d'un utilisateur dont le courriel est connu.
try{
    Utilisateur utilisateur= cerbere.findUtilisateursByMel(courriel);
    if( utilisateur != null) {
        // OK trouvé ...
    } else {
        // non trouvé ...
    }
  }
} catch (CerbereException e) {
    // Traitement en cas d'exception.
}
```

5 - Classes de l'interface de programmation Cerbère.

(Ce diagramme est fourni en annexe dans la documentation)



■