

[Personal](#) [Open source](#) [Business](#) [Explore](#)[Pricing](#) [Blog](#) [Support](#)[This repository](#)[Sign in](#)[Sign up](#)[AlexandreFenyo](#) / [france-connect](#)[Watch](#) 1[Star](#) 0[Fork](#) 0[Code](#)[Issues](#) 0[Pull requests](#) 0[Pulse](#)[Graphs](#)Branch: [master](#)[france-connect](#) / [README.md](#)[Find file](#)[Copy path](#)

AlexandreFenyo Update README.md

3f42c5e 10 hours ago

1 contributor

1417 lines (1057 sloc) 110 KB

[Raw](#)[Blame](#)[History](#)

## Table of Contents *generated with DocToc*

- **KIF : Kit d'Intégration à FranceConnect**
  - Introduction
  - Configuration
    - Fichiers de configuration
    - Paramètres de configuration
      - Paramètres pour la fonction KIF-SP (POC de fournisseur de services)
        - Activation de la fonction
        - Configuration des endpoints
        - Configuration de la relation de confiance mutuelle avec FranceConnect
        - Configuration du mécanisme de déconnexion
        - Configuration du comportement du fournisseur de services
      - Paramètres pour la fonction KIF-IdP (Identity Provider)
    - Fichier de paramétrage
    - Configuration des traces
  - Exploitation des traces
    - Format des traces
      - Traces d'informations
      - Traces d'erreurs
    - Gestion des erreurs
      - Phase d'authentification
        - Comportement attendu
        - Session expirée
        - État invalide
        - Code d'autorisation invalide
        - Nonce invalide
        - Autres cas d'erreur d'authentification
      - Phase de déconnexion
  - Cinématique d'authentification
  - Cinématique de déconnexion
  - Intégration de MITREid Connect dans Spring
  - Création de nouvelles ressources
    - Répertoires
    - Ressources statiques
    - Pages jsp
    - Contrôleur et vues
  - Déploiement
    - Prérequis
    - Fichier pom pour Maven
    - Démarrage dans un serveur Tomcat embarqué
    - Démarrage dans un serveur Jetty embarqué

- Démarrage dans Eclipse
- Points d'attention avec Eclipse
- Goals Maven
- Passage en production
  - Chiffrement des flux
  - Équilibrage de charge et haute disponibilité
    - Multi-centres de production
    - Mono-centre de production
- Intégration rapide d'une application existante avec KIF-IdP
  - Introduction
  - Messages échangés
  - Endpoint
  - Cinématique
  - Exemple d'application
  - Fonction de cryptographie
  - Représentation textuelle d'un message chiffré
  - Représentation binaire d'un message en clair
  - Création d'une requête d'authentification
  - Création d'une réponse à une requête
  - Traitement d'une réponse
  - Vérifications de sécurité
  - Sources de l'application exemple
  - Configuration
  - Traces
  - Déploiement
- Support

## KIF : Kit d'Intégration à FranceConnect

---

### Introduction

---

Ce produit a deux objectifs :

1. fournir un exemple complet d'implémentation d'un fournisseur **FranceConnect** de niveau *production-grade* et correctement documenté :
  - en environnement **JEE**, à l'aide du framework **Spring Security**
  - en s'appuyant sur **MITREid Connect**, l'implémentation de référence d'**OpenID Connect** développée par le **MIT**
  - en intégrant le mécanisme de déconnexion, la gestion des traces et celle des erreurs d'authentification
2. permettre l'intégration facile de l'authentification FranceConnect dans une application existante :
  - quelle que soit la technologie utilisée (JEE, Ruby on Rails, Perl/CGI, PHP, Node.js, etc.)
  - en raccordant cette application à l'IdP (*Identity Provider*) interne de KIF, qui se charge d'implémenter la cinématique d'interfaçage avec FranceConnect en se présentant comme un fournisseur de services.

KIF est donc à la fois un POC (*proof of concept*) de fournisseur de services FranceConnect en environnement JEE et un IdP relai entre une application existante (*legacy application*) et l'IdP FranceConnect.

L'implémentation de la fonction POC est dénommée **KIF-SP** (*Service Provider*) et l'implémentation de la fonction relai est dénommée **KIF-IdP** (*Identity Provider*).

Ce produit est accessible en ligne, en IPv4 et en IPv6, afin de vous permettre de le tester sans avoir à l'installer au préalable :

- KIF-SP : <https://fenyo.net/kif-sp>
- KIF-IdP : <https://fenyo.net/fc-online>

Une image Docker intégrant KIF-SP et KIF-IdP est disponible sur Docker Hub :

 <https://hub.docker.com/r/fenyoa/franceconnect-kif/>

Elle vous permet de démarrer rapidement une expérimentation à l'aide de votre propre instance de KIF, sans avoir à préparer un environnement JEE+Tomcat ni la nécessité de devoir compiler l'application et la déployer vous-même. En quelques minutes, vous disposerez d'une solution opérationnelle déployée sur un de vos serveurs, ou chez un des fournisseurs de Cloud majeurs supportant Docker ([Amazon Web Services](#), [Google Cloud Platform](#), [Microsoft Azure](#), [RedHat OpenShift Online](#), [OVH](#), [Cloudwatt](#), ...).

## Configuration

### Fichiers de configuration



Tous les exemples de configuration supposent que l'utilisateur a lancé un navigateur sur le hôte de KIF-IdP, ce qui permet de contacter le fournisseur de services avec l'adresse de boucle locale 127.0.0.1. Cela simplifie le déploiement et les tests car cette adresse est valide sur tout système intégrant une pile TCP/IP. Néanmoins, cela impose que le navigateur utilisé pour les tests soit démarré sur le même hôte que le fournisseur de services, afin qu'il puisse le contacter en s'adressant à 127.0.0.1. En production, il faut remplacer 127.0.0.1 par le nom DNS du fournisseur de services. Les flux directs entre le fournisseur de services et FranceConnect (web-services REST) sont par défaut chiffrés à l'aide de SSL/TLS. Par contre, les connexions au fournisseur de services sont configurées par défaut pour être réalisées via le protocole non chiffré HTTP, afin d'éviter d'imposer la mise en place d'un certificat de sécurité X.509 sur le fournisseur de services. En production, il faut remplacer ce protocole http dans les URL de configuration du fournisseur de services par https. Cela permet de chiffrer les flux entre le navigateur et le fournisseur de services avec SSL/TLS. Ce chiffrement est essentiel car ces flux contiennent des secrets, comme le cookie de session par exemple.

Deux fichiers de configuration sont utilisés :

- le fichier de configuration des paramètres
- le fichier de configuration des traces

La configuration des paramètres consiste à créer le fichier de paramétrage `config.properties` dans le répertoire `FournisseurDeServices/src/main/webapp/META-INF` à partir du modèle `config.properties-template` déjà présent dans ce même répertoire.



Le démarrage de l'application n'est pas possible avant d'avoir réalisé la configuration des paramètres car le fichier `config.properties` est référencé par le descripteur de déploiement d'application (*web application deployment descriptor*) `web.xml`.

Ce fichier de configuration contient des secrets, il faut donc configurer ses permissions d'accès ainsi que les permissions globales d'accès au système de fichiers sur lequel il est stocké de manière à ce qu'aucune personne non habilitée puisse y accéder.

La configuration des traces consiste à adapter le fichier `log4j.xml` (format standard [Log4j 1.2.x](#)) présent dans le répertoire `FournisseurDeServices/src/main/resources` aux besoins de traces. Dans ce même répertoire, deux exemples sont fournis :

- `log4j-devel.xml` avec un niveau de traces élevé, notamment sur les composants concernés par l'authentification
- `log4j-prod.xml` avec un niveau de trace peu verbeux, intégrant néanmoins toutes les traces nécessaires pour un système en production :
  - conservation des traces de niveau `warn` et supérieurs pour tous les composants
  - conservation des traces de niveau `info` et supérieurs pour KIF, incluant pour chaque événement de création ou suppression de session, d'authentification, d'erreur d'authentification ou de déconnexion (mécanisme de déconnexion global) :
    - l'identifiant de session (valeur du cookie JSESSIONID)
    - la référence à l'objet Java représentant la requête, permettant de croiser les traces correspondant à une même requête si besoin
    - l'adresse IP d'origine de la connexion TCP contenant le flux HTTP provenant du navigateur client
    - le port TCP côté client, pour différencier différents navigateurs présentant une adresse IP commune (cas où l'utilisateur emprunte un proxy ou un réseau NATé, par exemple)
    - la cause d'erreur le cas échéant (jeton invalide, tentative de rejou, etc.)
    - le nom de la session créée ou détruite le cas échéant
    - le nombre de sessions actives à des fins de *capacity planning* : la mémoire utilisée est une fonction affine du nombre de sessions

### Paramètres de configuration

## Paramètres pour la fonction KIF-SP (POC de fournisseur de services)

### Activation de la fonction

- `net.fenyo.franceconnect.config.oidc.debug`
  - type : booléen
  - valeur par défaut : `true` (fonction KIF-SP activée)
  - usage : activation/désactivation de la fonction KIF-SP (POC de fournisseur de service). Positionner la valeur `false` pour le passage en production de la fonction KIF-IdP (IdP relai), afin de désactiver l'exemple de fournisseur de services.

### Configuration des endpoints

Quatre endpoints sont déclarés pour la configuration de la cinématique d'authentification via FranceConnect : trois endpoints fournis par FranceConnect et un endpoint pour le fournisseur de service.

- `net.fenyo.franceconnect.config.oidc.authorizationendpointuri`
  - type : URL
  - valeur par défaut : <https://fcp.integ01.dev-franceconnect.fr/api/v1/authorize> (valeur utilisée par la plate-forme de développement/intégration de FranceConnect)
  - usage : authorization endpoint de FranceConnect, qui réalise l'authentification de l'utilisateur en le renvoyant vers un fournisseur d'identité FranceConnect et qui fournit en retour un authorization code au fournisseur de services.
- `net.fenyo.franceconnect.config.oidc.tokenendpointuri`
  - type : URL
  - valeur par défaut : <https://fcp.integ01.dev-franceconnect.fr/api/v1/token> (valeur utilisée par la plate-forme de développement/intégration de FranceConnect)
  - usage : token endpoint de FranceConnect, contacté directement par le fournisseur de services (invocation d'un web-service REST, donc sans passer par le navigateur de l'utilisateur) pour récupérer, en échange du code d'autorisation, un id token JWT et un access token. La signature de l'id token par FranceConnect est vérifiée par le fournisseur de service. Si cette signature est invalide ou si d'autres éléments de sécurité contenus dans ce jeton sont incorrects, l'authentification est rejetée et un message d'erreur du type suivant est ajouté dans le fichier de traces :  
`authentication failure exception:`  
`[org.springframework.security.authentication.AuthenticationServiceException: ...]` . Dans ce message d'erreur, la chaîne `...` est remplacée par la cause précise du rejet.
- `net.fenyo.franceconnect.config.oidc.userinfoendpointuri`
  - type : URL
  - valeur par défaut : <https://fcp.integ01.dev-franceconnect.fr/api/v1/userinfo> (valeur utilisée par la plate-forme de développement/intégration de FranceConnect)
  - usage : userinfo endpoint de FranceConnect, contacté directement par le fournisseur de services (invocation d'un web-service REST, donc sans passer par le navigateur de l'utilisateur) pour récupérer, en échange de l'access token, l'identité pivot de l'utilisateur (userinfo) au format JSON.
- `net.fenyo.franceconnect.config.oidc.redirecturi`
  - type : URL
  - valeur par défaut : [http://127.0.0.1/openid\\_connect\\_login](http://127.0.0.1/openid_connect_login)
  - usage : URL du endpoint de callback du fournisseur de services. URL où l'utilisateur est renvoyé après déconnexion du service, qu'il ait accepté ou pas la déconnexion de FranceConnect. **Le choix de la chaîne `openid_connect_login` est imposé par l'implémentation MITREid Connect, elle ne doit donc pas être substituée par une autre chaîne.** Cette URL est le endpoint fournisseur de services de MITREid Connect, lui permettant de recevoir le code d'autorisation et d'enchaîner alors la cinématique de récupération des jetons et de l'identité de l'utilisateur. **Cette URL doit être déclarée par le fournisseur de services sur le [portail de configuration FranceConnect](#) dans la section "Urls de callback".**

### Configuration de la relation de confiance mutuelle avec FranceConnect

- `net.fenyo.franceconnect.config.oidc.clientid`
  - type : chaîne de caractères (représentation hexadécimale d'un nombre de 256 bits)
  - valeur par défaut : aucune

- usage : client id du fournisseur de services, attribué par FranceConnect sur le [portail d'inscription](#). Cet identifiant est public.
- `net.fenyo.franceconnect.config.oidc.clientsecret`
  - type : chaîne de caractères (représentation hexadécimale d'un nombre de 256 bits)
  - valeur par défaut : aucune
  - usage : secret id du fournisseur de services, attribué par FranceConnect sur le [portail d'inscription](#). Ce secret partagé ne doit pas être divulgué. Pour cette raison, les droits d'accès au fichier `config.properties` doivent être configurés de telle façon que seul le conteneur d'application puisse accéder à son contenu.
- `net.fenyo.franceconnect.config.oidc.issuer`
  - type : chaîne de caractères
  - valeur par défaut : <https://fcp.integ01.dev-franceconnect.fr> (valeur utilisée par la plate-forme de développement/intégration de FranceConnect)
  - usage : identifiant de l'émetteur des id token JWT, attendu dans le claim (au sens 'revendication') *iss* de ces jetons. Si le claim reçu ne correspond pas à la valeur attendue, l'authentification est rejetée et le message d'erreur suivant est ajouté dans le fichier de traces : `authentication failure exception: [org.springframework.security.authentication.AuthenticationServiceException: Issuers do not match]`.
- `net.fenyo.franceconnect.config.oidc.fcbuttonuri`
  - type : URL
  - valeur par défaut : <https://fcp.integ01.dev-franceconnect.fr/js/franceconnect.js>
  - usage : URL du code JavaScript du bouton FranceConnect.

#### Configuration du mécanisme de déconnexion

- `net.fenyo.franceconnect.config.oidc.logouturi`
  - type : URL
  - valeur par défaut : <https://fcp.integ01.dev-franceconnect.fr/api/v1/logout> (valeur utilisée par la plate-forme de développement/intégration de FranceConnect)
  - usage : URL de déconnexion globale (*global logout*). Quand l'utilisateur souhaite se déconnecter du fournisseur de service, ce dernier invalide sa session puis le redirige vers cette URL chez FranceConnect, afin qu'il puisse aussi choisir de se déconnecter de FranceConnect. Il est ensuite redirigé vers le portail du fournisseur de services. Il est à noter que si l'utilisateur fait le choix de se connecter de FranceConnect, il ne sera pas pour autant déconnecté des autres fournisseurs de services auxquels il pourrait être simultanément connecté (hormis le fournisseur de services par lequel il a initié la demande de déconnexion). Ce mécanisme correspond donc un "global logout" partiel.
- `net.fenyo.franceconnect.config.oidc.afterlogouturi`
  - type : URL
  - valeur par défaut : <http://127.0.0.1/>
  - usage : URL où l'utilisateur est renvoyé après déconnexion du service, qu'il ait accepté ou non la déconnexion de FranceConnect. Cette URL ne pointe pas forcément sur le fournisseur de services, elle peut potentiellement correspondre au site institutionnel associé. Cette URL est aussi utilisée en cas d'erreur d'authentification, pour proposer à l'utilisateur de retourner au site institutionnel. **Cette URL doit être déclarée par le fournisseur de services sur le [portail de configuration FranceConnect](#) dans la section "Urls de redirection de déconnexion".**
- `net.fenyo.franceconnect.config.oidc.startlogouturi`
  - type : URL
  - valeur par défaut : `j_spring_security_logout`
  - usage : URL de logout utilisée par le bouton FranceConnect ou le fournisseur de services pour initier la séquence de déconnexion.

#### Configuration du comportement du fournisseur de services

- `net.fenyo.franceconnect.config.oidc.sessiontimeout`
  - type : nombre (minutes)
  - valeur par défaut : 240 minutes (4 heures)
  - usage : sans activité pendant ce délai, la session expire donc l'accès à une page protégée nécessite une nouvelle authentification via FranceConnect. Si cette valeur est inférieure à la durée de session de FranceConnect (30

minutes), la reconnexion pourra être transparente dans certains cas.

Exemple de séquence de reconnexion transparente :

- `sessiontimeout` vaut 10 minutes
  - l'utilisateur se connecte au fournisseur de services et s'authentifie via FranceConnect à  $t_0$
  - à partir de  $t_0 + 5 \text{ min}$ , l'utilisateur devient inactif
  - sa session chez le fournisseur de services est donc invalide à partir de  $t_0 + 5 \text{ min} + \text{sessiontimeout}$ , c'est-à-dire  $t_0 + 15 \text{ min}$
  - à  $t_0 + 20 \text{ min}$ , l'utilisateur reprend son activité en accédant à une page protégée
  - la session ayant expiré, le fournisseur de services renvoie l'utilisateur s'authentifier chez FranceConnect
  - la session FranceConnect n'ayant pas expiré (si l'utilisateur n'a pas réalisé une déconnexion via le bouton FranceConnect entre-temps, depuis ce fournisseur de services ou un autre), FranceConnect fournit un jeton d'autorisation au fournisseur de services sans interaction utilisateur
  - le fournisseur de services utilise ce jeton d'autorisation pour récupérer l' id token et l'identité de l'utilisateur
- `net.fenyo.franceconnect.config.oidc.authenticationerroruri`
    - type : URL
    - valeur par défaut : <http://127.0.0.1/authenticationError>
    - usage : URL où l'utilisateur est renvoyé en cas d'erreur d'authentification. Si cette URL pointe vers `/authenticationError` sur le fournisseur de service, l'utilisateur se verra alors proposé de continuer sa navigation sur l'URL définie par `net.fenyo.franceconnect.config.oidc.afterlogouturi`.

#### Paramètres pour la fonction KIF-IdP (Identity Provider)

- `net.fenyo.franceconnect.config.idp.key`
  - type : clé AES 256 bits en hexadecimal (generation via openssl : `openssl rand -hex 32`)
  - valeur par défaut : `a6a7ee7abe681c9c4cede8e3366a9ded96b92668ea5e26a31a4b0856341ed224`
  - usage : secret partagé permettant le chiffrement entre l'application à intégrer et KIF-IdP.
- `net.fenyo.franceconnect.config.idp.iv`
  - type : vecteur d'initialisation 128 bits en hexadecimal (generation via openssl : `openssl rand -hex 16`)
  - valeur par défaut : `87b7225d16ea2ae1f41d0b13fdce9bba`
  - usage : vecteur d'initialisation nécessaire pour le chiffrement.
- `net.fenyo.franceconnect.config.idp.redirecturi`
  - type : URL
  - valeur par défaut : `https://fenyo.net/fc/identite.cgi?`
  - usage : début de l'URL de callback de l'application.

Les valeurs par défaut permettent de communiquer avec l'application exemple disponible sur <https://fenyo.net/fc/index.cgi>

#### Fichier de paramétrage

Voici un exemple complet de fichier de paramétrage `config.properties` :

```
# pour la fonction KIF-FS (POC de fournisseur de services)
net.fenyo.franceconnect.config.oidc.debug=true
net.fenyo.franceconnect.config.oidc.clientid=a100f6923ae93e2e5a8e2e2fb1d0c1e988d8c06c43a8ef838e64eeb932750405
net.fenyo.franceconnect.config.oidc.clientsecret=f7cf2c8cf9863e009c3a520b5f380d9849adad21fe21c0e2782bc9cb25908a67
net.fenyo.franceconnect.config.oidc.sessiontimeout=240
net.fenyo.franceconnect.config.oidc.issuer=https://fcp.integ01.dev-franceconnect.fr
net.fenyo.franceconnect.config.oidc.authorizationendpointuri=https://fcp.integ01.dev-franceconnect.fr/api/v1/authorize
net.fenyo.franceconnect.config.oidc.tokenendpointuri=https://fcp.integ01.dev-franceconnect.fr/api/v1/token
net.fenyo.franceconnect.config.oidc.userinfoendpointuri=https://fcp.integ01.dev-franceconnect.fr/api/v1/userinfo
net.fenyo.franceconnect.config.oidc.redirecturi=http://127.0.0.1/openid_connect_login
net.fenyo.franceconnect.config.oidc.logouturi=https://fcp.integ01.dev-franceconnect.fr/api/v1/logout
net.fenyo.franceconnect.config.oidc.afterlogouturi=http://127.0.0.1/
net.fenyo.franceconnect.config.oidc.authenticationerroruri=http://127.0.0.1/authenticationError
net.fenyo.franceconnect.config.oidc.startlogouturi=j_spring_security_logout
net.fenyo.franceconnect.config.oidc.fcbbuttonuri=https://fcp.integ01.dev-franceconnect.fr/js/franceconnect.js
# pour la fonction KIF-IdP (IdP relais)
net.fenyo.franceconnect.config.idp.key=a6a7ee7abe681c9c4cede8e3366a9ded96b92668ea5e26a31a4b0856341ed224
net.fenyo.franceconnect.config.idp.iv=87b7225d16ea2ae1f41d0b13fdce9bba
net.fenyo.franceconnect.config.idp.redirecturi=https://fenyo.net/fc/identite.cgi?
```

## Configuration des traces

Voici un exemple complet de fichier de configuration des traces `log4j.xml` pour un serveur en production, les traces étant fournies dans la sortie standard du serveur d'application :

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE log4j:configuration PUBLIC "-//APACHE//DTD LOG4J 1.2//EN" "log4j.dtd">
<log4j:configuration xmlns:log4j="http://jakarta.apache.org/log4j/">

  <!-- Appenders -->
  <appender name="console" class="org.apache.log4j.ConsoleAppender">
    <param name="Target" value="System.out" />
    <layout class="org.apache.log4j.PatternLayout">
      <param name="ConversionPattern" value="%-5p: %d %c - %m%n" />
    </layout>
  </appender>

  <!-- Application Loggers -->
  <logger name="net.fenyo">
    <level value="info" />
  </logger>

  <!-- Root Logger -->
  <root>
    <priority value="warn" />
    <appender-ref ref="console" />
  </root>

</log4j:configuration>
```

Fournir les traces sur la sortie standard du serveur d'application n'est pas toujours le moyen le plus adapté pour les conserver. C'est un moyen idéal dans le cadre d'un serveur d'application hébergé dans un micro-conteneur [Docker](#), mais dans la plupart des autres situations, on peut remplacer l'appender de type `ConsoleAppender` pour les stocker plutôt :

- dans un fichier, que l'on fait tourner régulièrement : avec un appender de type [DailyRollingFileAppender](#)
- sur un serveur SYSLOG centralisé, avec un appender de type [SyslogAppender](#)
- dans un fichier d'événements Windows potentiellement rerouté vers un [SIEM](#) comme [LogRhythm](#), avec un appender de type [NTEventLogAppender](#)

## Exploitation des traces

### Format des traces

#### Traces d'informations

À chaque création de session, une trace correspondante est générée, incluant la valeur du nouvel identifiant de session stocké dans le cookie JSESSIONID et une mise à jour du nombre de sessions en mémoire :

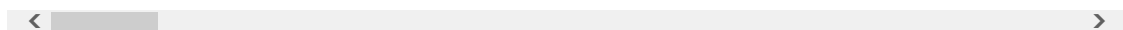
```
INFO : 2016-07-24 03:25:16,203 net.fenyo.franceconnect.SessionListener - log fc: msg [création de session 2236652D2F6
```



À chaque invalidation de session, une trace correspondante est générée, incluant l'identifiant de session supprimé, l'identité de l'utilisateur concerné et une mise à jour du nombre de sessions en mémoire (il n'y a pas d'origine de requête car la session peut être supprimée par l'expiration du timeout de sessions, sans action utilisateur) :

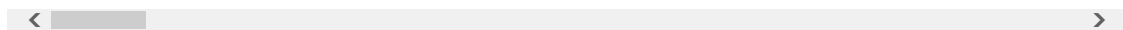
Exemple de session supprimée à l'expiration du timeout de session :

```
INFO : 2016-07-24 04:21:14,525 net.fenyo.franceconnect.SessionListener - log fc: msg [destruction de session 2236652C
```



Exemple de session supprimée à la demande d'un utilisateur (l'adresse IP et le port TCP d'origine de la requête sont fournis dans la trace *logout* qui suit la trace *destruction de session* ; ces deux traces peuvent être croisées car elles disposent toutes deux du même identifiant de session 2236652D2F06E535395CDA6CA557B9BF) :

```
INFO : 2016-07-24 03:36:56,092 net.fenyo.franceconnect.SessionListener - log fc: msg [destruction de session 2236652C
INFO : 2016-07-24 03:36:56,095 net.fenyo.franceconnect.LogoutHandler - log fc: msg [logout]; auth: id token [eyJ0eXAiOi
```



À chaque accès à une ressource protégée, une trace correspondante est générée, incluant notamment l'identité de l'utilisateur, l'identifiant de session, une référence à la requête, l'adresse IP et le port d'origine de la requête :

```
INFO : 2016-07-24 03:25:33,139 net.fenyo.franceconnect.WebController - log fc: msg [accès à /user]; auth: id token [e
```

## Traces d'erreurs

Lorsqu'une erreur liée à l'authentification se produit, une trace correspondante est générée :

```
INFO : 2016-07-24 03:47:24,138 net.fenyo.franceconnect.AuthenticationFailureHandler - log fc: msg [authentication fai
```

## Gestion des erreurs

### Phase d'authentification

#### Comportement attendu

Le comportement standard de MitreID Connect, en cas d'erreur d'authentification, consiste à indiquer au navigateur une erreur de type 401, incluant un descriptif de la cause de l'erreur. Dans KIF, ce comportement a été remplacé par la bonne pratique de sécurité consistant à cacher les précisions concernant la cause de l'erreur d'authentification. La cause de l'erreur est tracée et l'utilisateur est renvoyé vers une page d'erreur générique qu'on peut positionner par le paramètre de configuration `net.fenyo.franceconnect.config.oidc.authenticationerroruri`.

#### Session expirée

Si la session a expiré entre l'envoi vers FranceConnect et le retour avec le code d'autorisation, alors une [trace d'erreur](#) est générée avec le message suivant : "Authentication Failed: State parameter mismatch on return. Expected null got 2f3e7b5c97c0c". La valeur null indique que l'état associé à la session (paramètre `state` dans le protocole OpenID Connect) n'a pas pu être trouvé car il n'y a pas de session ou que cette session n'a jamais tenté de se connecter via FranceConnect. L'utilisateur est alors redirigé vers la page d'erreur définie par le paramètre de configuration `net.fenyo.franceconnect.config.oidc.authenticationerroruri`. Si la valeur de ce paramètre est une URL qui pointe vers `/authenticationError` sur le fournisseur de service, l'utilisateur se verra alors proposé de continuer sa navigation sur l'URL définie par la valeur du paramètre `net.fenyo.franceconnect.config.oidc.afterlogouturi`.

#### État invalide

Si l'état (paramètre `state` dans le protocole OpenID Connect) ne correspond pas à celui envoyé à FranceConnect dans le cadre de l'authentification de cette session, alors une [trace d'erreur](#) est générée avec le message suivant : "Authentication Failed: State parameter mismatch on return. Expected 3f3222875114b got 2f3e7b5c97c0c". La valeur attendue (3f3222875114b) est celle de l'état envoyé à FranceConnect et le faux état reçu est 2f3e7b5c97c0c. L'utilisateur est alors redirigé vers la page d'erreur définie par le paramètre de configuration `net.fenyo.franceconnect.config.oidc.authenticationerroruri`. Si la valeur de ce paramètre est une URL qui pointe vers `/authenticationError` sur le fournisseur de service, l'utilisateur se verra alors proposé de continuer sa navigation sur l'URL définie par la valeur du paramètre `net.fenyo.franceconnect.config.oidc.afterlogouturi`.

#### Code d'autorisation invalide

Si le code d'autorisation utilisé est faux ou a déjà été utilisé, alors l'échange suivant se produit avec FranceConnect :

- le fournisseur de services émet la requête suivante au token endpoint de FranceConnect :

```
POST /api/v1/token HTTP/1.1
Accept: text/plain, application/json, application/*+json, */*
Content-Type: application/x-www-form-urlencoded
Content-Length: 312
Host: fcp.integ01.dev-franceconnect.fr
Accept-Encoding: gzip,deflate

grant_type=authorization_code&code=1660c04e70db2b5311e6a7ab80c19246c3b7f123354d48c05f40d2aac3fb6c7c&redirect_uri=
```

- FranceConnect signale que le code est invalide :

```
HTTP/1.1 400 Bad Request
Server: nginx
```



```
Date: Wed, 20 Jul 2016 17:09:24 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 27
Connection: keep-alive
ETag: W/"1b-BTGn9J/xQnk2eWB3zdcJSA"
Vary: Accept-Encoding
```

- Une [trace d'erreur](#) est générée avec le message suivant : Authentication Failed: Unable to obtain Access Token: 400 Bad Request . L'utilisateur est alors redirigé vers la page d'erreur définie par le paramètre de configuration `net.fenyo.franceconnect.config.oidc.authenticationerroruri` . Si la valeur de ce paramètre est une URL qui pointe vers `/authenticationError` sur le fournisseur de service, l'utilisateur se verra alors proposé de continuer sa navigation sur l'URL définie par la valeur du paramètre `net.fenyo.franceconnect.config.oidc.afterlogouturi` .

#### Nonce invalide

Si le nonce reçu n'est pas celui attendu, une [trace d'erreur](#) est générée avec un message indiquant une possible tentative d'attaque par rejeu (*replay*). L'utilisateur est alors redirigé vers la page d'erreur définie par le paramètre de configuration `net.fenyo.franceconnect.config.oidc.authenticationerroruri` . Si la valeur de ce paramètre est une URL qui pointe vers `/authenticationError` sur le fournisseur de service, l'utilisateur se verra alors proposé de continuer sa navigation sur l'URL définie par la valeur du paramètre `net.fenyo.franceconnect.config.oidc.afterlogouturi` .

#### Autres cas d'erreur d'authentification

De nombreuses vérifications de sécurité sont imposées par le protocole OpenID Connect, par exemple la vérification de la signature des id token JWT. Si une de ces vérifications conduit à une erreur, une [trace d'erreur](#) est générée avec un message décrivant la raison de cette erreur. L'utilisateur est alors redirigé vers la page d'erreur définie par le paramètre de configuration `net.fenyo.franceconnect.config.oidc.authenticationerroruri` . Si la valeur de ce paramètre est une URL qui pointe vers `/authenticationError` sur le fournisseur de service, l'utilisateur se verra alors proposé de continuer sa navigation sur l'URL définie par la valeur du paramètre `net.fenyo.franceconnect.config.oidc.afterlogouturi` .

#### Phase de déconnexion

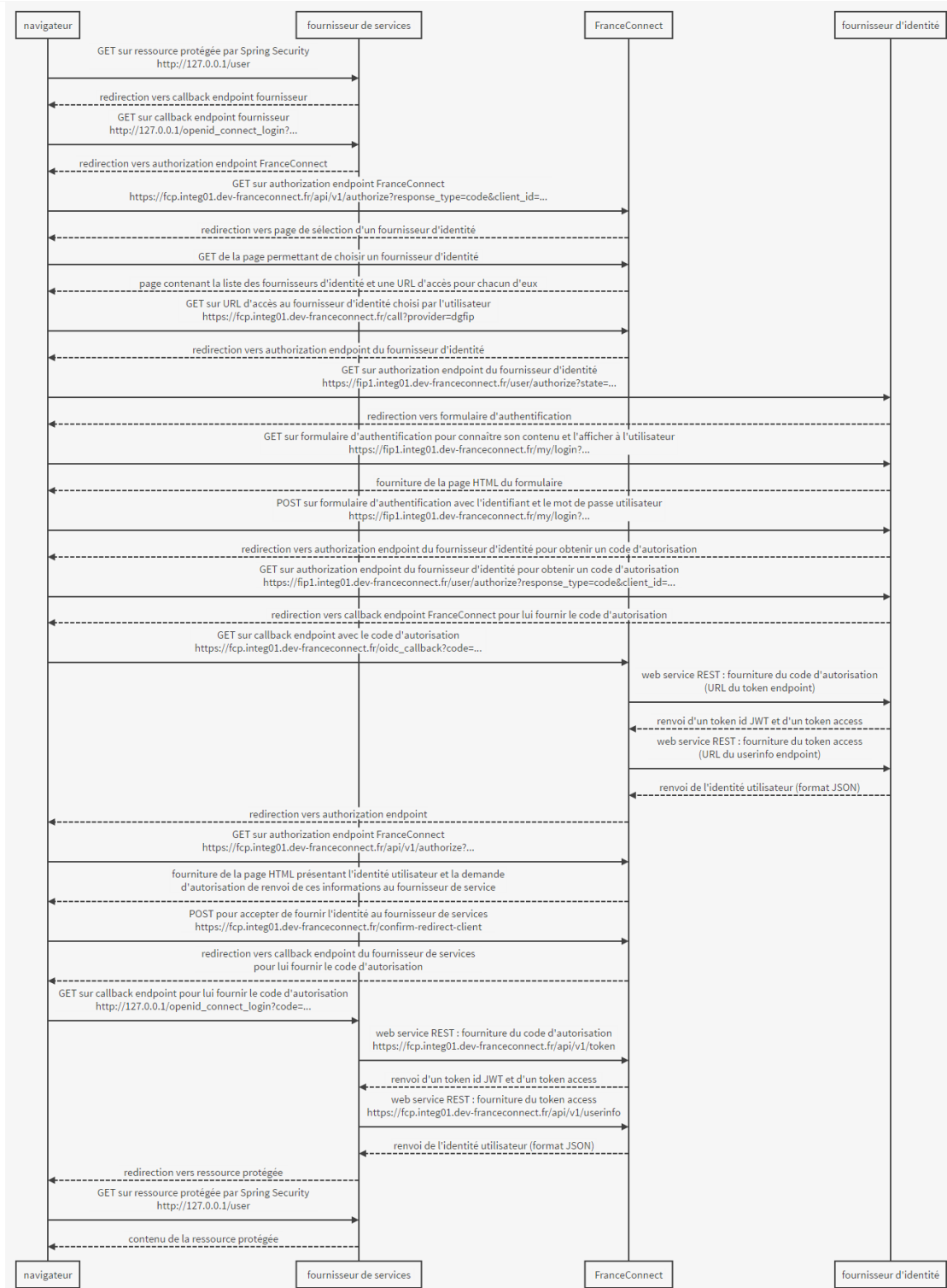
Au moment d'une tentative de déconnexion, si la session a déjà expiré ou si une déconnexion s'est déjà produite, le fournisseur de services redirige le navigateur vers l'URL configurée après déconnexion sans passer par FranceConnect puisqu'il ne dispose plus d'id token à lui indiquer. Ce cas se produit par exemple si deux onglets sont ouverts sur l'application et sont connectés via la même session, et que l'un des onglets a réalisé une déconnexion via le bouton FranceConnect. Si une déconnexion est alors initiée par le bouton FranceConnect du second onglet, il n'y a plus de contexte d'authentification, donc on ne peut ni ne doit renvoyer vers FranceConnect pour une déconnexion.

## Cinématique d'authentification

La cinématique d'authentification est constituée des étapes suivantes :

1. Lorsque le filtre Spring MITREid Connect détecte l'accès à une ressource protégée et qu'il n'y a pas eu de précédente authentification pour la session courante, MITREid Connect redirige alors l'utilisateur vers son endpoint de callback.
2. Ce endpoint constate qu'aucun id token n'est associé à cette session et qu'aucun paramètre contenant un code d'autorisation n'est fourni dans la requête qu'il reçoit.
3. Il entame donc la cinématique OpenID Connect pour demander un code d'autorisation à l'authorization endpoint et ce code est renvoyé par FranceConnect sur ce endpoint.
4. À la réception du code, le endpoint de callback invoque alors un web-services REST vers le token endpoint de FranceConnect pour récupérer un id token et un access token.
5. Un nouveau web-service REST présentant l'access token est invoqué sur le userinfo endpoint de FranceConnect pour récupérer le userinfo qui représente l'identité de l'utilisateur au format JSON.
6. L'utilisateur est enfin renvoyé vers la ressource protégée, à laquelle il a désormais accès.

Ce diagramme de séquence UML présente l'ensemble des échanges en jeu dans cette phase d'authentification entre les différents acteurs (navigateur, fournisseur de services, FranceConnect, fournisseur d'identité). Les informations d'état civil de l'utilisateur, renvoyées par le fournisseur d'identité, sont redressées par FranceConnect pour constituer une identité dite pivot, à l'aide d'un rapprochement avec le contenu du Répertoire national d'identification des personnes physiques ([RNIPP](#)). L'identité fournie par le fournisseur d'identité n'est donc pas forcément celle relayée au fournisseur de services par FranceConnect : **le fournisseur de services reçoit systématiquement un extrait de l'état civil provenant du RNIPP**.



## Cinématique de déconnexion

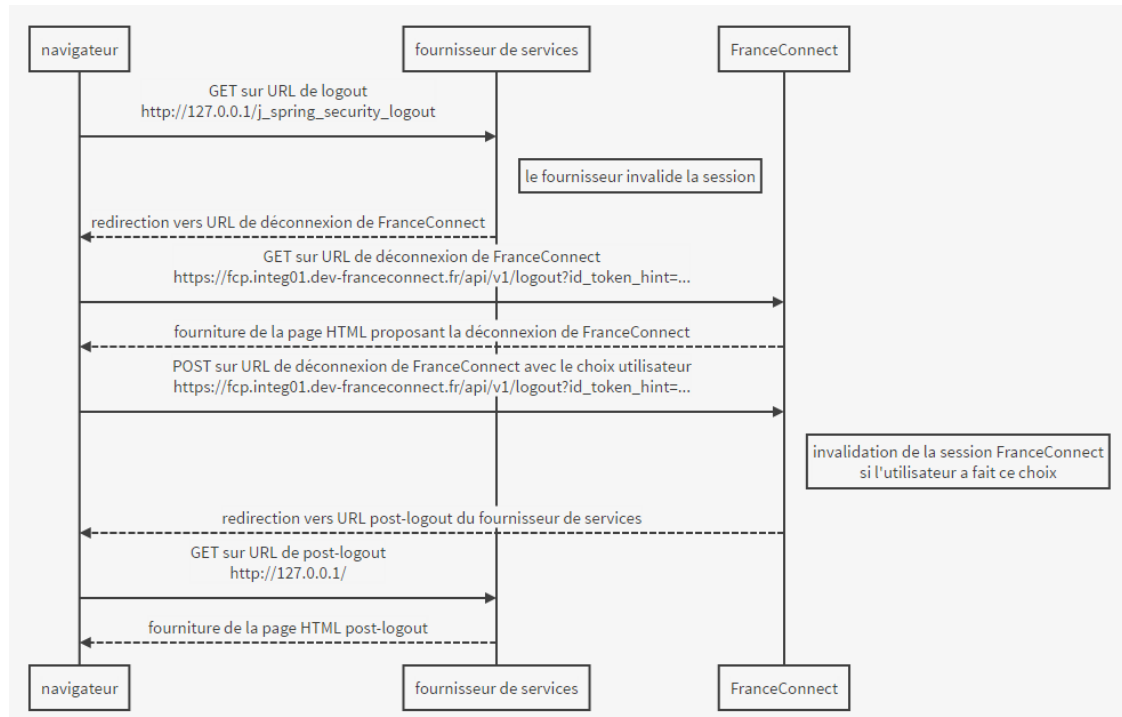
La cinématique de déconnexion n'est pas spécifiée dans le protocole OpenID Connect, mais directement par FranceConnect. C'est pour cela que l'on n'évoque pas ici des endpoints mais simplement des URL de logout (déconnexion) et de post-logout. N'étant pas une cinématique normée, elle n'est pas implémentée par MitreID Connect et a donc dû être développée spécifiquement dans KIF-SP.

La cinématique de déconnexion est constituée des étapes suivantes :

1. Lorsque l'utilisateur clique sur le lien de déconnexion dans le bouton FranceConnect ou via un lien d'une page du fournisseur de services, son navigateur effectue une requête GET sur l'URL de logout du fournisseur de services.
2. Le contrôleur Spring MVC invalide alors la session et redirige le navigateur vers l'URL de logout de FranceConnect.
3. FranceConnect propose à l'utilisateur de se déconnecter de FranceConnect, ou de rester connecté.

4. Selon le choix de l'utilisateur, FranceConnect invalide éventuellement la session utilisateur et, dans tous les cas, redirige son navigateur vers l'URL post-logout du fournisseur de services.
5. Le fournisseur de services fournit la page HTML correspondant à son URL post-logout.

Ce diagramme de séquence UML présente l'ensemble des échanges en jeu dans cette phase de déconnexion où interviennent le navigateur, le fournisseur de services et FranceConnect (le fournisseur d'identité n'est pas concerné, ni informé de la déconnexion) :



## Intégration de MITREid Connect dans Spring

MITREid Connect est un filtre de sécurité [Spring Security](#). L'intégration de l'authentification OpenID Connect dans KIF-SP a donc consisté à configurer [Spring MVC](#) pour s'appuyer sur [Spring Security](#) afin de protéger les ressources et à configurer un filtre [Spring Security](#) à l'aide de MITREid Connect.

Cette configuration a été mise en place dans le fichier décrivant la servlet [Spring MVC](#), `franceconnect-servlet.xml` :

```

<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:mvc="http://www.springframework.org/schema/mvc"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:tx="http://www.springframework.org/schema/tx"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:security="http://www.springframework.org/schema/security"
    xmlns:oauth="http://www.springframework.org/schema/security/oauth2"
    xmlns:util="http://www.springframework.org/schema/util"
    xsi:schemaLocation="http://www.springframework.org/schema/security/oauth2 http://www.springframework.org/schem
        http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-4.1.xsd
        http://www.springframework.org/schema/security http://www.springframework.org/schema/security/spring-secu:
        http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.1.x:
        http://www.springframework.org/schema/util http://www.springframework.org/schema/util/spring-util-4.1.xsd
        http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.1.xsd
        http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context:

    <!-- permettre la configuration via des annotations sous net.fenyo.franceconnect -->
    <mvc:annotation-driven />
    <context:component-scan base-package="net.fenyo.franceconnect" />

    <!-- importer les valeurs des paramètres de configuration de l'accès à FranceConnect -->
    <context:property-placeholder location="META-INF/config.properties" />

    <!-- mappings d'URI directs, sans nécessiter de passer par un contrôleur -->
    <mvc:resources mapping="/static/**" location="/static/" />
    <mvc:resources mapping="/js/**" location="/js/" />
    <mvc:resources mapping="/css/**" location="/css/" />
  
```

```

<mvc:resources mapping="/html-noauth/**" location="/html-noauth/" />
<mvc:resources mapping="/html/**" location="/html-noauth/" />
<mvc:resources mapping="/jsp-noauth/**" location="/jsp-noauth/" />

<!-- mappings directs dont les URI sont déclarées, à la fin de ce fichier de configuration, comme nécessitant u
<mvc:resources mapping="/html-needauth/**" location="/html-needauth/" />
<mvc:resources mapping="/jsp-needauth/**" location="/jsp-needauth/" />

<!-- indiquer à Spring MVC comment résoudre l'emplacement des vues à partir de leur nom -->
<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name="prefix" value="/WEB-INF/views/" />
    <property name="suffix" value=".jsp" />
</bean>

<!-- injecter automatiquement les informations d'authentification dans le contexte -->
<mvc:interceptors>
    <bean class="org.mitre.openid.connect.web.UserInfoInterceptor" />
</mvc:interceptors>

<!-- signaler à Spring Security d'utiliser un authentication manager contenant un authentication provider qui e
<!-- OIDCAuthenticationProvider se charge de contacter le userinfo endpoint avec l'autorization bearer pour re
<security:global-method-security pre-post-annotations="enabled" proxy-target-class="true" authentication-manage

<bean id="openIdConnectAuthenticationProvider" class="org.mitre.openid.connect.client.OIDCAuthenticationProvide

<security:authentication-manager alias="authenticationManager">
    <security:authentication-provider ref="openIdConnectAuthenticationProvider" />
</security:authentication-manager>

<!-- création du authRequestUrlBuilder, utilisé par le filtre de pré-authentification MITREid Connect fourni à
<bean class="org.mitre.openid.connect.client.service.impl.PlainAuthRequestUrlBuilder" id="plainAuthRequestUrlBu

<!-- création d'un bean servant à stocker l'URI correspondant au fournisseur d'identité (provider représentant
<bean class="org.mitre.openid.connect.client.service.impl.StaticSingleIssuerService" id="staticIssuerService">
    <property name="issuer" value="${net.fenyo.franceconnect.config.oidc.issuer}" />
</bean>

<!-- création d'un bean stockant les trois enpoints du fournisseur d'identité FranceConnect (https://fcp.integ
<bean class="org.mitre.openid.connect.client.service.impl.StaticServerConfigurationService" id="staticServerCor
    <property name="servers">
        <map>
            <entry key="${net.fenyo.franceconnect.config.oidc.issuer}">
                <bean class="org.mitre.openid.connect.config.ServerConfiguration">
                    <property name="issuer" value="${net.fenyo.franceconnect.config.oidc.issuer}" />
                    <property name="authorizationEndpointUri" value="${net.fenyo.franceconnect.config.oidc.authorization
                    <property name="tokenEndpointUri" value="${net.fenyo.franceconnect.config.oidc.tokenendpointuri}" />
                    <property name="userInfoUri" value="${net.fenyo.franceconnect.config.oidc.userinfoendpointuri}" />
                </bean>
            </entry>
        </map>
    </property>
</bean>

<!-- création d'un bean stockant les paramètres et le endpoint du fournisseur de service, ce bean étant fourni
<bean class="org.mitre.openid.connect.client.service.impl.StaticClientConfigurationService" id="staticClientCor
    <property name="clients">
        <map>
            <entry key="${net.fenyo.franceconnect.config.oidc.issuer}">
                <bean class="org.mitre.oauth2.model.RegisteredClient">

                    <property name="scope">
                        <set value-type="java.lang.String">
                            <value>openid</value>
                            <value>gender</value>
                            <value>birthdate</value>
                            <value>birthcountry</value>
                            <value>birthplace</value>
                            <value>given_name</value>
                            <value>family_name</value>
                            <value>email</value>
                            <value>address</value>
                            <value>preferred_username</value>
                            <value>phone</value>
                            <!-- <value>profile</value> -->
                        </set>
                    </property>

                    <property name="tokenEndpointAuthMethod" value="SECRET_POST" />

```

```

        <property name="clientId" value="${net.fenyo.franceconnect.config.oidc.clientid}" />
        <property name="clientSecret" value="${net.fenyo.franceconnect.config.oidc.clientsecret}" />

        <!-- l'URI de redirection est imposée par MITREid Connect (en dur dans le code source de MITREid Connect)
        <!-- cette configuration ne sert qu'aux validations de sécurité -->
        <property name="redirectUri">
            <set>
                <value>${net.fenyo.franceconnect.config.oidc.redirecturi}</value>
            </set>
        </property>

    </bean>
</entry>
</map>
</property>
</bean>

<!-- URI vers lesquelles l'utilisateur est redirigé en cas d'erreur d'authentification -->
<bean id="authenticationFailureHandler" class="net.fenyo.franceconnect.AuthenticationFailureHandler">
    <property name="defaultFailureUrl" value="${net.fenyo.franceconnect.config.oidc.authenticationerroruri}" />
    <property name="exceptionMappings">
        <props>
            <prop key="org.springframework.security.authentication.AuthenticationServiceException">${net.fenyo.francecon
        </props>
    </property>
</bean>

<!-- création d'un filtre de pré-authentification Spring Security implémenté par MITREid Connect et configuré en ré
<bean id="openIdConnectAuthenticationFilter" class="org.mitre.openid.connect.client.OIDCAuthenticationFilter">
    <property name="authenticationManager" ref="authenticationManager" />
    <property name="issuerService" ref="staticIssuerService" />
    <property name="serverConfigurationService" ref="staticServerConfigurationService" />
    <property name="clientConfigurationService" ref="staticClientConfigurationService" />
    <property name="authRequestUrlBuilder" ref="plainAuthRequestUrlBuilder" />
    <!-- propriété propre à Spring Security, proposée par la classe parente de OIDCAuthenticationFilter : org.springf
        le comportement en cas d'erreur est défini par le bean authenticationFailureHandler, défini précédemment -->
    <property name="authenticationFailureHandler" ref="authenticationFailureHandler" />
</bean>

<!-- création d'un bean servant à stocker l'URI correspondant au point d'entrée d'authentification via Spring Secur
    il s'agit du callback endpoint du fournisseur de services, qui reçoit le code d'autorisation -->
<bean id="authenticationEntryPoint" class="org.springframework.security.web.authentication.LoginUrlAuthenticationEr
    <constructor-arg value="${net.fenyo.franceconnect.config.oidc.redirecturi}" />
</bean>

<!-- implémentation de la séquence de déconnexion FranceConnect -->
<bean id="logoutHandler" class="net.fenyo.franceconnect.LogoutHandler">
    <property name="logoutUri" value="${net.fenyo.franceconnect.config.oidc.logouturi}" />
    <property name="afterLogoutUri" value="${net.fenyo.franceconnect.config.oidc.afterlogouturi}" />
</bean>

<!-- configuration de Spring Security
    note : l'attribut entry-point-ref sert à déclarer le point d'entrée d'authentification en cas d'accès à une UR
<security:http auto-config="false" use-expressions="true" disable-url-rewriting="true" entry-point-ref="authenticat

    <!-- déclaration du filtre MITREid Connect -->
    <security:custom-filter before="PRE_AUTH_FILTER" ref="openIdConnectAuthenticationFilter" />

    <!--
    configuration de Spring Security avec :
    - l'URI de logout utilisée par le bouton FranceConnect ou le fournisseur de services pour initier la séquence de
    - le bean à invoquer après le logout effectif de l'application : ce bean se charge d'implémenter la cinématique
      - redirection de l'utilisateur chez FranceConnect, qui lui propose aussi de se déloguer de FranceConnect,
      - retour vers le fournisseur de service
    -->
    <security:logout success-handler-ref="logoutHandler" logout-url="${net.fenyo.franceconnect.config.oidc.startlogc
    <!-- FranceConnect réalise le logout via GET et non POST ; pour que Spring Security le supporte, il faut désactiv
    <security:csrf disabled="true" />

    <!-- déclaration des URI nécessitant une authentification valide -->
    <security:intercept-url pattern="/html-needauth/*" access="isFullyAuthenticated()" />
    <security:intercept-url pattern="/jsp-needauth/*" access="isFullyAuthenticated()" />

</security:http>
</beans>

```

## Création de nouvelles ressources

## Répertoires

Consultez les mappings au début du fichier `franceconnect-servlet.xml` afin de connaître les sous-répertoires de `FournisseurDeServices/src/main/webapp/` où rajouter :

- des fichiers statiques, des pages html, du code JavaScript, des pages de styles CSS et des images accessibles sans qu'ils ne nécessitent une authentification au préalable
- des fichiers statiques, des pages html, du code JavaScript, des pages de styles CSS et des images accessibles uniquement après une authentification préalable

En cas d'accès à une ressource protégée mais sans authentification préalable de l'utilisateur, la cinématique d'authentification est automatiquement lancée puis le navigateur est redirigé à nouveau vers la ressource à laquelle il peut enfin accéder.

## Ressources statiques

Les accès aux ressources statiques (pages html, code JavaScript, pages de styles CSS et images) nécessitant une authentification ne sont **pas** tracés. Ces ressources n'ont pas accès à l'identité de l'utilisateur.

## Pages jsp

Pour tracer l'accès direct à une page jsp, c'est-à-dire sans passer par le contrôleur Spring MVC, celle-ci doit inclure `<% net.fenyo.franceconnect.Tools.log(request.getRequestURI()); %>`.

Les pages jsp à accès direct sans passer par un contrôleur n'ont pas accès à l'identité de l'utilisateur (néanmoins, cet accès est possible en écrivant du code Java spécifique dans ces pages).

## Contrôleur et vues

Le contrôleur a accès à l'identité de l'utilisateur. Les vues, c'est-à-dire les pages jsp accessible après passage par le contrôleur, ont aussi accès aux données d'identité de l'utilisateur.

Pour créer un mapping de requête au sein du contrôleur, il faut modifier la classe `net.fenyo.franceconnect.WebController` en y ajoutant une méthode spécifique. Le mapping est configuré par une annotation `@RequestMapping`, par exemple :

`@RequestMapping(value = "/user", method = RequestMethod.GET)`. Si la requête nécessite une authentification, il faut le signaler par l'annotation suivante : `@PreAuthorize("isFullyAuthenticated()")`. Pour tracer cette requête, la méthode spécifique doit débiter par une invocation de la méthode statique `log` de la classe `Tools`, par exemple : `Tools.log("accès à /user", logger);`. La variable `logger` est une variable d'instance du contrôleur.

Voici un exemple de méthode spécifique qui mappe l'accès à `/user` et invoque la vue `user` :

```
@RequestMapping(value = "/user", method = RequestMethod.GET)
@PreAuthorize("isFullyAuthenticated()")
public ModelAndView user(final Principal p) {
    Tools.log("accès à /user", logger);
    final ModelAndView mav = new ModelAndView("user");

    // On injecte dans le modèle les champs de userinfo qui ne sont pas dans le standard OpenID Connect
    // mais néanmoins transportés dans les identités FranceConnect, car la variable userinfo
    // automatiquement insérée dans le modèle par MitreID Connect ne contient pas de getter
    // pour ces champs. Il s'agit uniquement de birthplace et birthcountry.
    final OIDCAuthenticationToken oidcauth = (OIDCAuthenticationToken) SecurityContextHolder.getContext().getAuth
mav.addObject("oidcBirthplace", oidcauth.getUserInfo().getSource().get("birthplace"));
mav.addObject("oidcBirthcountry", oidcauth.getUserInfo().getSource().get("birthcountry"));

    // on injecte dans le modèle les paramètres de configuration, notamment pour le bouton FranceConnect
    mav.addObject("oidcAttributes", oidcAttributes);

    return mav;
}
```

La vue a accès à l'identité de l'utilisateur via la variable `userInfo` automatiquement injectée dans le modèle (mais cette variable ne contient pas les champs `birthplace` ni `birthcountry` : consulter l'exemple de code ci-dessus pour leur traitement particulier). Elle doit inclure le bouton `FranceConnect`, notamment pour permettre la déconnexion de l'utilisateur.

Voici un exemple de vue affichant les informations d'identité de l'utilisateur :

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>

<html lang="fr">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>Fournisseur de services FranceConnect</title>
</head>

<body>
  <!-- inclusion du code JavaScript de FranceConnect.
       Ce code force le navigateur à récupérer la CSS suivante : https://fcp.integ01.dev-franceconnect.fr/styleshee
       Cette CSS définit le attributs de style pour l'élément d'id fconnect-profile -->
  <script src="${ oidcAttributes.fcbbuttonuri }"></script>
  <!-- inclusion du bouton FranceConnect -->
  <div style="color: #000000; background-color: #000ccc" id="fconnect-profile" data-fc-logout-url="${ oidcAttribute
  <a href="#">${ userInfo.givenName } ${ userInfo.familyName }&nbsp;<i class="material-icons tiny">keyboard_arrow_c

  Cette page de fourniture du service n'est accessible qu'aux utilisateurs authentifiés.
  Vous êtes <b>correctement authentifié</b> via FranceConnect.<br/>

<pre>
Utilisateur (user info) :
- sujet (utilisateur) : ${ userInfo.sub }
- genre : ${ userInfo.gender }
- date de naissance : ${ userInfo.birthdate }
- prénom : ${ userInfo.givenName }
- nom : ${ userInfo.familyName }
- courriel : ${ userInfo.email }
- adresse postale :
  - rue : ${ userInfo.address.streetAddress }
  - commune : ${ userInfo.address.locality }
  - région : ${ userInfo.address.region }
  - code postal : ${ userInfo.address.postalCode }
  - pays : ${ userInfo.address.country }
  - lieu de naissance : ${ oidcBirthplace }
  - pays de naissance : ${ oidcBirthcountry }
Valeur JSON complète : ${ userInfo.source }
</pre>
</body>
</html>
```

Une méthode spécifique du contrôleur a la possibilité de ne pas imposer une authentification préalable. Dans ce cas, la vue peut être construite de manière à activer ou désactiver des blocs selon que l'authentification a été réalisée ou non. Pour cela, il faut utiliser la bibliothèque de tags de Spring Security et encadrer les blocs nécessitant une authentification dans un élément `<authorize access="isFullyAuthenticated()">` et ceux à afficher uniquement en l'absence d'authentification dans un élément `<authorize access="!isFullyAuthenticated()">`. Le bouton FranceConnect doit être inclus uniquement lors d'un accès authentifié.

Voici un exemple de méthode spécifique correspondant à ce scénario :

```
@RequestMapping(value = "/user", method = RequestMethod.GET)
public ModelAndView user(final Principal p) {
    Tools.log("accès à /user", logger);
    final ModelAndView mav = new ModelAndView("user");
    // on injecte dans le modèle les paramètres de configuration, notamment pour le bouton FranceConnect
    mav.addObject("oidcAttributes", oidcAttributes);
    return mav;
}
```

Voici la vue associée :

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib prefix="security" uri="http://www.springframework.org/security/tags" %>
<%@ page session="false" %>

<html lang="fr">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>Fournisseur de services FranceConnect</title>
</head>

<body>
  <security:authorize access="isFullyAuthenticated()">
```

```

<!-- inclusion du code JavaScript de FranceConnect.
    Ce code force le navigateur à récupérer la CSS suivante : https://fcp.integ01.dev-franceconnect.fr/stylesh
    Cette CSS définit le attributs de style pour l'élément d'id fconnect-profile -->
<script src="{ oidcAttributes.fbuttonuri }"></script>
<!-- inclusion du bouton FranceConnect -->
<div style="color: #000000; background-color: #000ccc" id="fconnect-profile" data-fc-logout-url="{ oidcAttribu
<a href="#">${ userInfo.givenName } ${ userInfo.familyName }&nbsp;<i class="material-icons tiny">keyboard_arrow

</security:authorize>

<H1>Page d'accueil du service</H1>
Cette page d'accueil du service est accessible à tous les internautes, authentifiés ou non.
<HR/>

<security:authorize access="isFullyAuthenticated()">
    Vous vous êtes précédemment <b>correctement authentifié</b> auprès du fournisseur de services via FranceConnect
</security:authorize>

<security:authorize access="!isFullyAuthenticated()">
    Vous n'êtes <b>pas</b> authentifié à ce fournisseur de services.<br/>
</security:authorize>
</body>
</html>

```

## Déploiement

### Prérequis

Voici la liste des prérequis nécessaires à l'utilisation opérationnelle de ce produit :

- disposer d'un environnement Unix, Linux ou Windows
- installer un environnement JDK 7 ou version supérieure
- installer Maven 3.0.4 ou version supérieure
- disposer d'une connexion Internet (accès direct ou par proxy)
- disposer d'un navigateur Internet

### Fichier pom pour Maven

Les dépendances implicites lors de l'intégration d'un composant Java dans un application JEE conduisent souvent, si l'on n'y prend pas garde, à des conflits entre versions. En effet, il arrive souvent que deux composants nécessaires à l'application s'appuient sur des versions distinctes d'un même autre composant. Dans un tel cas, il est préférable d'imposer le choix de la version retenue afin d'éviter un comportement incohérent de l'application. Le fichier `pom.xml` a été spécifiquement écrit dans cette optique, après une analyse des conflits de versions, et les versions des composants retenus sont les suivantes :

groupe	artefact	version
org.springframework	spring-core	4.2.5
org.springframework.security	spring-security-config	4.0.4
org.springframework.security.oauth	spring-security-oauth2	2.0.9
org.mitre	openid-connect-client	1.2.6
org.slf4j	slf4j-api	1.7.21
javax.servlet	servlet-api	2.5
javax.servlet.jsp	jsp-api	2.1
javax.servlet	jstl	1.2
junit	junit	4.12
org.slf4j	slf4j-api	1.7.21
org.slf4j	jcl-over-slf4j	1.7.21
com.fasterxml.jackson.core	jackson-annotations	2.3.4
org.apache.commons	commons-lang3	3.4
org.apache.httpcomponents	httpclient	4.5.2



Les conflits potentiels qui sont résolus par ces choix de version concernaient les problématiques suivantes :

- MitreID Connect 1.2.x induit des dépendances directes vers la version 3 du framework Spring ainsi que certaines dépendances transitives issues de Spring (on appelle ici *dépendances transitives* les dépendances de dépendances). Or on utilise ici la version 4 de Spring, plus évoluée. On s'est donc affranchi de ces dépendances directes ou transitives, pour profiter pleinement de Spring framework 4. De même, les dépendances de MitreID Connect implique d'autres conflits avec certains composants utilisés par ailleurs par KIF. L'ensemble de ces conflits concernent précisément les artefacts Maven suivants de MitreID Connect, qu'on a donc exclus du traitement des dépendances par Maven :
  - org.springframework/spring-core
  - org.springframework/spring-webmvc
  - org.springframework.security/spring-security-core
  - org.springframework.security/spring-security-config
  - org.springframework.security.oauth/spring-security-oauth2
  - org.apache.httpcomponents/httpclient
  - org.slf4j/slf4j-api
  - org.slf4j/jcl-over-slf4j
  - com.fasterxml.jackson.core/jackson-annotations
- MitreID Connect s'appuie sur spring-context, ce dernier s'appuyant sur commons-logging. Or MitreID Connect s'appuie sur SLF4J en lieu et place de commons-logging. On exclut donc la dépendance de spring-context vers commons-logging.
- Spring OAuth 2.0.9 induit des dépendances transitives vers des composants anciens du framework Spring. On s'affranchit donc de ces dépendances, en les excluant du traitement Maven pour Spring OAuth :
  - org.springframework/spring-core
  - org.springframework/spring-webmvc
  - org.springframework/spring-context
  - org.springframework/spring-beans
  - org.springframework.security/spring-security-core
  - org.springframework.security/spring-security-config
  - org.springframework.security/spring-security-web
- Les versions 1.2.3 à 1.2.7 de MITREid Connect référencent la bibliothèque Bouncy Castle Crypto nommée bcprov-jdk15on. Sachant que cette bibliothèque est utilisée par MITREid Connect uniquement pour chiffrer et déchiffrer des jetons, mais pas pour les signer ou vérifier leur signature, cette bibliothèque est donc inutile dans le cadre des cinématiques FranceConnect. Or ce package induit des délais de recherche d'annotations importants (cf. <http://stackoverflow.com/questions/17584495/unable-to-complete-the-scan-for-annotations-for-web-application-app-due-to-a>), pouvant conduire à un timeout au chargement de l'application sous Jetty. On évite donc l'importation de la version bcprov-jdk15on de cette bibliothèque. Ce phénomène n'apparaît pas jusqu'à la version 1.2.2 de MITREid Connect, car les références à la bibliothèque Bouncy Castle Crypto n'apparaissent qu'à partir de la version 1.2.3.
- On importe une version récente de Bouncy Castle Crypto n'impliquant pas des délais de recherche d'annotations importants, non pas pour utilisation par MITREid Connect, mais pour utilisation par l'implémentation d'un IdP dans ce package. Cet IdP est implémenté par la méthode idp() de la classe WebController (contrôleur Spring). Cet IdP n'est pas utile pour mettre en oeuvre la cinématique FranceConnect. Cette dépendance peut donc être supprimée à condition de supprimer aussi le code de l'IdP. Dans cet IdP, on utilise Bouncy Castle Crypto plutôt que l'implémentation native de Sun/Oracle via l'API JCE car cette dernière limite par défaut les clés AES à 128 bits alors qu'on souhaite utiliser une clé de 256 bits pour des raisons de force de chiffrement.

Le fichier pom.xml effectue plusieurs vérifications avant d'entamer un traitement :

- il vérifie explicitement qu'il est interprété avec Maven 3.0.4 ou version supérieure et dans le cas contraire, il interrompt le traitement Maven avec le message d'erreur suivant :

```
FRANCECONNECT - ERREUR DE CONFIGURATION

CE PACKAGE NECESSITE L'UTILISATION DE MAVEN 3.0.4 OU VERSION SUPERIEURE.
```

- il vérifie explicitement qu'il est interprété dans un environnement Java 1.7 ou version supérieure et dans le cas contraire, il interrompt le traitement Maven avec le message d'erreur suivant :

```
FRANCECONNECT - ERREUR DE VERSION MAVEN

CE PACKAGE NECESSITE L'UTILISATION D'UN JDK JAVA VERSION 1.7 AU MINIMUM.
```

- il vérifie explicitement que le fichier de paramétrage a été créé et dans le cas contraire, il interrompt le traitement Maven avec le message d'erreur suivant :

```
FRANCECONNECT - ERREUR DE CONFIGURATION

AVANT DE COMMENCER A UTILISER CE PACKAGE,
VOUS DEVEZ RECOPIER LE FICHIER src/main/webapp/META-INF/config.properties-template
DANS src/main/webapp/META-INF/config.properties
ET Y METTRE A JOUR LES VALEURS REELLES DE VOS IDENTIFIANTS FOURNISSEUR FRANCECONNECT.
```

## Démarrage dans un serveur Tomcat embarqué

- Démarrage avec Tomcat sur un serveur avec accès direct à Internet : `mvn clean tomcat7:run`
- Démarrage avec Tomcat sur un serveur nécessitant le passage par un proxy web pour accéder à Internet (remplacer PROXYHOST et PROXYPORT par les valeurs correspondant au proxy web) :
  - `mvn -Dhttps.proxyHost=PROXYHOST -Dhttps.proxyPort=PROXYPORT clean tomcat7:run`
  - attendre le message `INFO: Starting ProtocolHandler ["http-bio-80"]`, signifiant le lancement complet de Tomcat, avant de passer à l'étape suivante
  - lancer alors un navigateur sur <http://127.0.0.1/>
- Pour modifier le port local par défaut (80), au cas où il y aurait par exemple déjà un serveur sur ce port, ou au cas où l'utilisateur courant n'aurait pas les droits pour écouter sur un port privilégié (port dont le numéro est strictement inférieur à 1024), rajoutez l'option `-Dnet.fenyo.franceconnect.config.listen.port=PORT` sur la ligne de commande mvn (remplacer PORT par la valeur du port TCP local d'écoute souhaité).

## Démarrage dans un serveur Jetty embarqué

- Démarrage avec Jetty sur un serveur avec accès direct à Internet : `mvn clean jetty:run`
- Démarrage avec Jetty sur un serveur nécessitant le passage par un proxy web pour accéder à Internet (remplacer PROXYHOST et PROXYPORT par les valeurs correspondant au proxy web) :
  - `mvn -Dhttps.proxyHost=PROXYHOST -Dhttps.proxyPort=PROXYPORT clean jetty:run`
  - attendre le message `[INFO] Started Jetty Server`, signifiant le lancement complet de Jetty, avant de passer à l'étape suivante
- lancer alors un navigateur sur <http://127.0.0.1/>
- Pour modifier le port local par défaut (80), au cas où il y aurait par exemple déjà un serveur sur ce port, ou au cas où l'utilisateur courant n'aurait pas les droits pour écouter sur un port privilégié (port dont le numéro est strictement inférieur à 1024), rajoutez l'option `-Dnet.fenyo.franceconnect.config.listen.port=PORT` sur la ligne de commande mvn (remplacer PORT par la valeur du port TCP local d'écoute souhaité).

## Démarrage dans Eclipse

- Installer [Eclipse Neon](#) ou version supérieure (afin d'éviter de devoir installer certaines extensions spécifiques pour le support de composants comme Maven par exemple)
- Importer le projet dans [Eclipse](#)
- À chaque modification du fichier `pom.xml`, sélectionner l'élément racine du projet dans l'explorateur de projet, faire "bouton droite" sur cet élément, choisir *Maven* dans le menu déroulant puis *Update Project...*
- Pour (re-)compiler le projet, sélectionner dans le menu Projet l'entrée *Clean...* puis *Build Project*
- Pour démarrer l'application, utiliser le menu Run pour accéder à *Run configurations...* ou *Debug configurations...*, créer une configuration Apache Tomcat (vous devrez disposer d'une distribution Tomcat 7 ou version supérieure), publier l'application dans le serveur et démarrer le serveur

## Points d'attention avec Eclipse



KIF est configuré par défaut pour se déployer dans le contexte racine ( "/" ) du serveur d'application et non pas dans un contexte correspondant à un chemin intermédiaire comme `"/poc-franceconnect"`. [Eclipse](#) peut être amené à modifier le chemin de déploiement de l'application, ce qui empêche son bon fonctionnement car certaines des URL déclarées dans le fichier de configuration ne sont plus valables. Dans ce cas, il faut soit réécrire ces URL, soit repositionner correctement le chemin dans [Eclipse](#), comme ceci :

- soit interrompre Eclipse et modifier le fichier  
FournisseurDeServices/.settings/org.eclipse.wst.common.component pour que la ligne `<property name="context-root" value=""/>` contienne bien un attribut `value` vide (c'est cet attribut qui est parfois modifié par Eclipse). Puis redémarrer Eclipse.
- soit sélectionner *Show View* dans le menu déroulant Window puis *Servers*. Dans la vue *Servers* qui s'affiche, double-cliquer sur le serveur (ex.: Tomcat v8.0 Server at localhost [Stopped, Synchronized] ), la configuration du serveur s'affiche alors et l'onglet sélectionné par défaut est nommé Overview, changer pour l'onglet Modules et vérifier que la colonne Path du tableau des Web Modules chargés est vide. Si ce n'est pas le cas, cliquer sur la ligne correspondante et sur le bouton `Edit...` pour effacer le contenu du champ Path. Sauver la nouvelle configuration (entrée *Save* du menu Fichier, ou raccourci clavier `Ctrl-S` ) et redémarrer le serveur.



Le fichier de configuration `config.properties` contient deux URLs déclarées chez FranceConnect :

- l'URL de callback (paramètre `net.fenyo.franceconnect.config.oidc.redirecturi` de `config.properties` )
- l'URL de redirection de déconnexion (paramètre `net.fenyo.franceconnect.config.oidc.afterlogouturi` de `config.properties` )

Si ces URL utilisent l'adresse IP 127.0.0.1, le navigateur doit être lancé sur <http://127.0.0.1/>

Si ces URL utilisent le nom localhost, le navigateur doit être lancé sur <http://localhost/>

En effet, les cookies de session positionnés par un serveur désigné localhost ne sont pas renvoyés à un serveur désigné 127.0.0.1 et réciproquement. **Utiliser dans une même configuration un mélange de localhost et de 127.0.0.1 conduit à des erreurs de connexion.**

## Goals Maven

Voici une liste des goals Maven utilisés régulièrement :

- faire le ménage (supprimer le répertoire `target` ) : `mvn clean`
- compiler : `mvn clean compile`
- créer une archive war : `mvn clean package` (le fichier war créé se nomme `franceconnect-demo-1.0.0-BUILD.war` et se trouve dans le sous-répertoire `target` )

## Passage en production

### Chiffrement des flux

En production, il faut remplacer 127.0.0.1 par le nom DNS du fournisseur de services dans le fichier de configuration `franceconnect-servlet.xml` . Il faut remplacer le protocole http dans les URL de configuration du fournisseur de services par https. Cela permet de chiffrer les flux entre le navigateur et le fournisseur de services avec SSL/TLS. Ce chiffrement est essentiel car ces flux contiennent des secrets, comme le cookie de session par exemple. Pour que les flux soient effectivement chiffrés, la pratique habituelle consiste à faire porter l'adresse IP associée au nom public du fournisseur de services par un reverse-proxy de type Apache ou Nginx, ou par un équilibreur de flux, points de terminaison des sessions SSL/TLS. C'est alors le reverse-proxy ou l'équilibreur de charge qui est doté d'un certificat X.509 et de la clé privée associée. Les flux entre le reverse-proxy ou l'équilibreur de charge et le fournisseur de services restent non chiffrés. Les invocations, par le fournisseur de services, de web-services REST vers FranceConnect sont chiffrés car les endpoints FranceConnect sont accessibles uniquement via SSL/TLS.

### Équilibrage de charge et haute disponibilité

#### Multi-centres de production

Plusieurs instances de KIF peuvent être déployées simultanément pour la montée en charge et/ou la haute-disponibilité. Si le serveur d'application a été configuré en cluster, afin que les données de sessions des serveurs de servlets soient répliquées entre les différents hôtes hébergeant KIF, le partage de charges multi-centres de production est possible via un simple round-robin DNS (exemple de configuration avec Tomcat 8 : <https://tomcat.apache.org/tomcat-8.0-doc/cluster-howto.html>). Avec une telle architecture, aucun équilibreur de charge n'est requis, sauf si on veut rajouter la fonction failover pour laquelle une paire d'équilibreurs (une paire en fonctionnement actif/passif suffit, pour éviter que l'équilibreur soit un SPOF : *Single Point Of Failure*) de type GSLB (Global Server Load Balancing) est nécessaire, comme par exemple la solution d'[équilibrage F5 BIG-IP munie du module GTM \(Global Traffic Management\)](#). Le principe de fonctionnement de ce type d'équilibreur consiste à gérer dynamiquement le contenu d'une zone DNS contenant des enregistrements à durée de vie courte (quelques secondes) pour associer le nom global du service aux adresses IP des instances du cluster de servlets constituant le fournisseur de services. La

zone est gérée par plusieurs serveurs faisant autorité (*authoritative name servers*) afin que le service DNS ne constitue pas un SPOF (*Single Point Of Failure*).

#### Mono-centre de production

Plusieurs instances de KIF peuvent être déployées simultanément pour la montée en charge et/ou la haute-disponibilité sans nécessiter la mise en place d'un mécanisme de mise en cluster par réplication des données de session. Les données d'une session étant alors présentes dans une seule instance de KIF, un équilibreur (ou plutôt une paire en fonctionnement actif/passif) disposé en coupure des flux doit implémenter le mécanisme d'affinité de session, cette affinité étant basée uniquement sur la valeur du cookie de session JSESSIONID.

## Intégration rapide d'une application existante avec KIF-IdP

### Introduction

KIF-IdP, inclus dans KIF, est une implémentation d'un fournisseur d'identité (IdP) permettant une intégration simple d'une application existante :

- quelle que soit la technologie utilisée (JEE, Ruby on Rails, Perl/CGI, PHP, Node.js, etc.),
- sans impacter le mode d'authentification existant, afin que le fournisseur de services puisse continuer à authentifier lui-même les utilisateurs qui ne souhaitent pas s'authentifier via FranceConnect,
- en raccordant cette application à l'IdP (*Identity Provider*) interne de KIF, qui se charge d'implémenter la cinématique d'interfaçage avec FranceConnect en se présentant comme un fournisseur de services.

L'application existante est le fournisseur de services, mais au lieu de s'appuyer directement sur FranceConnect en tant que fournisseur d'identité, elle s'appuie sur KIF-IdP. Celui-ci relaie les demandes d'autorisation ou de déconnexion vers FranceConnect. L'application existante est donc déchargée de l'implémentation du protocole OpenID Connect, de l'invocation de web-services REST chez FranceConnect et de la capacité à vérifier des signatures de jetons JWT.

La relation de confiance entre l'application existante et KIF-IdP est établie à l'aide d'un mécanisme de chiffrement AES-256-CBC. Ces deux entités se partagent donc une clé AES de 256 bits et un vecteur d'authentification de 128 bits.

### Messages échangés

Deux types de messages chiffrés sont échangés entre l'application et KIF-IdP, par l'intermédiaire du navigateur, en exploitant le mécanisme de redirection :

- de l'application vers KIF-IdP : les requêtes d'authentification, qui sont constituées d'une URL de callback, qui peut contenir différents paramètres, et doit contenir, parmi ceux-ci, les deux suivants, présents une seule fois chacun :
  - `state` : un paramètre opaque représentant de manière unique la session utilisateur (il peut s'agir par exemple de la valeur du cookie de session, si les sessions sont associées à un tel élément dans le cadre de la technologie utilisée pour le développement de l'application, de la valeur d'une donnée associée uniquement à la session sans être transportée dans un cookie, ou encore du résultat de l'application d'une fonction de hash d'une de ces valeurs, comme SHA-256)
  - `nonce` : un code aléatoire opaque permettant d'éviter les attaques par jeu

Chacun de ces deux paramètres doit être composé d'une chaîne de caractères répondant aux exigences normatives lui permettant d'être transportée dans un paramètre d'une URL (cf. [RFC-1738](#)).

- de KIF-IdP vers l'application : les réponses aux précédentes requêtes, contenant l'identité pivot ainsi que les valeurs des paramètres `state` et `nonce` de la requête associée, au format JSON avec encodage UTF-8

On peut noter, pour confirmer la cohérence de ce mode d'échange, que les critères normatifs permettant d'être transporté dans un paramètre d'une URL impliquent qu'un encodage UTF-8 est aussi possible (la réciproque n'étant pas vraie).

### Endpoint

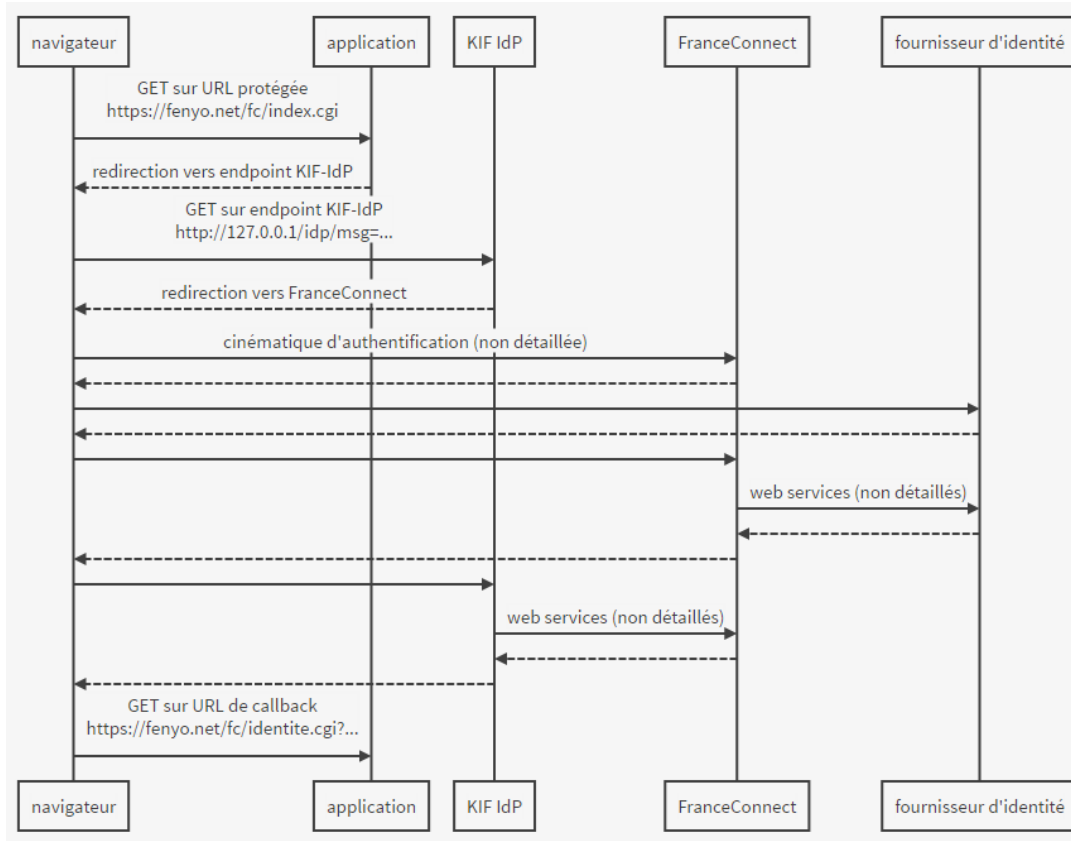
Ce protocole n'utilise qu'un seul endpoint : celui de KIF-IdP, qui lui permet de recevoir les requêtes d'authentification en provenance de l'application. Les réponses à ces requêtes ne sont pas envoyées vers un endpoint de l'application mais vers une URL de redirection incluse dans la requête de l'application.

Ce protocole est donc particulièrement léger à implémenter côté application puisque cette dernière n'a aucun endpoint à gérer.

L'URL relative du endpoint de KIF-IdP est `/idp`. Si KIF-IdP est par exemple lancé sur une hôte nommé `kif-idp.mon-domaine-institutionnel.fr` protégé par un reverse proxy qui termine les sessions SSL/TLS, alors le endpoint de KIF-IdP sera accessible à l'URL suivante : <https://kif-idp.mon-domaine-institutionnel.fr/idp>

## Cinématique

La cinématique d'authentification, qui est présentée en détails dans la suite de ce document, est résumée dans le diagramme de séquence UML suivant :



## Exemple d'application

Dans le cadre des exemples qui suivent, on suppose que l'utilisateur a lancé un navigateur sur le hôte de KIF-IdP, il pourra donc accéder au endpoint de KIF-IdP avec l'URL suivante : <http://127.0.0.1/idp>

L'application utilisée avec ces exemples est déployée à l'URL suivante : <https://fenyo.net/fc>

Elle est développée en langage shell, déployée via l'interface CGI sur un serveur apache et est constituée de trois scripts :

- une page d'accueil : <https://fenyo.net/fc/index.cgi>
- une page protégée : <https://fenyo.net/fc/identite.cgi>
- une page de déconnexion : <https://fenyo.net/fc/logout.cgi>

**Au total, ces trois scripts comptabilisent moins de 30 lignes de shell**, en incluant la gestion du bouton FranceConnect. Ils permettent de démontrer la facilité d'intégration d'une application à FranceConnect, en tant que fournisseur de services, à l'aide de KIF-IdP.

## Fonction de cryptographie

La fonction de cryptographie sur laquelle s'appuient l'application et KIF-IdP pour chiffrer les messages qu'ils échangent est AES-256-CBC (avec padding PKCS#7 par des blocs de 128 bits).

Ce chiffrement symétrique AES-256-CBC est donc caractérisé par :

- l'utilisation d'une clé secrète de 256 bits
- avec un padding préalable de type CBC
- en utilisant une taille de bloc de 128 bits (PKCS#7)
- donc avec un vecteur d'initialisation de 128 bits

On peut par exemple utiliser openssl pour chiffrer un message en clair à l'aide de cette fonction cryptographique :

```
% KEY=a6a7ee7abe681c9c4cede8e3366a9ded96b92668ea5e26a31a4b0856341ed224
% IV=87b7225d16ea2ae1f41d0b13fdce9bba
% echo Texte en clair | openssl aes-256-cbc -K $KEY -iv $IV > contenu-chiffre.bin
%
```

On peut constater, en utilisant od, que le message est bien devenu illisible :

```
% od -xa contenu-chiffre.bin
0000000  3c6b  814e  5d18  71a4  a11d  e828  9435  9ad1
          k  <  N soh can  ]  $  q  gs  !  (  h  5 dc4  Q sub
0000020
```

Pour le déchiffrer, on peut aussi utiliser openssl, comme ceci :

```
% KEY=a6a7ee7abe681c9c4cede8e3366a9ded96b92668ea5e26a31a4b0856341ed224
% IV=87b7225d16ea2ae1f41d0b13fdce9bba
% openssl aes-256-cbc -d -K $KEY -iv $IV < contenu-chiffre.bin
Texte en clair
%
```

## Représentation textuelle d'un message chiffré

Un message chiffré par AES-256-CBC est constitué d'une chaîne d'octets. Sa représentation sous forme d'une chaîne de caractères est la représentation hexadécimale de ces octets, en utilisant des minuscules et en s'appuyant sur la table de correspondances (aussi dénommée *charset* ou *character set*) US-ASCII. Il s'agit donc d'une chaîne de caractères dont la taille est le double de la taille de la chaîne d'octets initiale.



La représentation textuelle d'un message chiffré peut donc être passée en paramètre d'une URL sans nécessiter de transformation particulière puisqu'une URL ne contient que des caractères de la table de correspondances US-ASCII : cf. [RFC-1738](#).

On peut aussi noter que les charsets UTF-8 (utilisé mondialement), ISO-8859-1 (utilisé essentiellement pour les langues latines) et ISO-8859-15 (utilisé essentiellement en Europe) sont des sur-ensembles du charset US-ASCII. La représentation textuelle d'un message chiffré est donc identique dans ces trois charsets et dans le charset US-ASCII. N'importe quelle bibliothèque de fonctions capable d'utiliser l'un ou l'autre de ces charsets est donc capable de transformer un message chiffré vers sa représentation textuelle, et réciproquement.

On peut, par exemple, utiliser hexdump pour convertir un message chiffré vers sa représentation textuelle :

```
% hexdump -v -e '1/1 "%02x"' < contenu-chiffre.bin | read HEXA
% echo $HEXA
6b3c4e81185da4711da128e83594d19a
%
```

À l'inverse, pour convertir le message en hexadecimal vers sa forme binaire, on peut utiliser Perl5 :

```
% echo -n $HEXA | perl -pe 's/([0-9a-f]{2})/chr hex $1/gie' > contenu-chiffre-2.bin
% diff -s contenu-chiffre.bin contenu-chiffre-2.bin
Les fichiers contenu-chiffre.bin et contenu-chiffre-2.bin sont identiques
%
```

## Représentation binaire d'un message en clair

La fonction de cryptographie choisie n'agit que sur des messages binaires, constitués de chaînes d'octets. Les messages textuels échangés doivent donc pouvoir être transformés en binaire avant chiffrement et, réciproquement, les messages chiffrés doivent pouvoir être transformés en messages textuels après déchiffrement.

- Une requête d'authentification est une URL. Elle est donc formée d'une suite des caractères du charset US-ASCII. On choisit la représentation binaire canonique constituée d'une chaîne d'octets directement issue de ce charset. La taille de la représentation binaire est donc identique à celle de la requête en clair.

- Une réponse est un message JSON avec encodage UTF-8. On choisit une représentation binaire constituée d'une chaîne d'octets construite à partir de ce charset. Le nombre d'octets de la représentation binaire est donc supérieur ou égal au nombre de caractères constituant la réponse en clair, et est inférieur ou égal à quatre fois ce nombre de caractères.

## Création d'une requête d'authentification

Lorsque l'application existante souhaite effectuer une authentification via FranceConnect, elle procède aux étapes suivantes :

- Elle construit une URL de callback permettant à KIF-IdP de renvoyer l'utilisateur vers l'application existante après une authentification réussie. Cette URL de callback peut contenir différents paramètres, en incluant, parmi ceux-ci, les paramètres `state` et `nonce`, comme indiqué précédemment.

Voici un exemple d'URL de callback de ce type :

```
https://fenyo.net/fc/identite.cgi?nonce=2ff22cb9663990d009fd0dfe87d997c6&state=f894bb7061a7c2a2
```

- La chaîne de caractères représentant l'URL est alors transformée en une représentation binaire, par le mécanisme indiqué précédemment.

Voici un exemple d'une telle représentation binaire :

```
% echo -n "https://fenyo.net/fc/identite.cgi?nonce=2ff22cb9663990d009fd0dfe87d997c6&state=f894bb7061a7c2a2" > ur:
% od -xa url.bin
0000000 7468 7074 3a73 2f2f 6566 796e 2e6f 656e
          h t t p s : / / f e n y o . n e
0000020 2f74 6366 692f 6564 746e 7469 2e65 6763
          t / f c / i d e n t i t e . c g
0000040 3f69 6f6e 636e 3d65 6632 3266 6332 3962
          i ? n o n c e = 2 f f 2 2 c b 9
0000060 3636 3933 3039 3064 3930 6466 6430 6566
          6 6 3 9 9 0 d 0 0 9 f d 0 d f e
0000100 3738 3964 3739 3663 7326 6174 6574 663d
          8 7 d 9 9 7 c 6 & s t a t e = f
0000120 3938 6234 3762 3630 6131 6337 6132 0032
          8 9 4 b b 7 0 6 1 a 7 c 2 a 2
%
< >
```

- La chaîne d'octets correspondant à la représentation binaire est chiffrée avec le mécanisme AES-256-CBC, en utilisant la clé secrète et le vecteur d'initialisation partagés entre l'application et KIF-IdP, afin de produire le message chiffré.

Voici un exemple de message chiffré de cette manière :

```
% openssl aes-256-cbc -K $KEY -iv $IV < url.bin > contenu-chiffre.bin
% od -x contenu-chiffre.bin
0000000 6116 030e c4c5 a506 be9c fab6 3c49 8d7e
0000020 5f0a b99d c41b 3c6d 664f 0700 6b81 612f
0000040 ea15 6e2c 249e ff33 924e 3629 4178 be6d
0000060 8246 e66e ffc4 da23 f1a0 4e8c 1d11 f7d4
0000100 920f d7ac 3ebe 0767 03fd 8b21 4bea 32ce
0000120 d1ab 45ba afad 9dc0 3040 8ab2 74e6 2824
0000140
%
```

- La représentation textuelle du message chiffré est alors produite par le mécanisme indiqué précédemment.

Voici un exemple de cette représentation textuelle :

```
% hexdump -v -e '1/1 "%02x"' < contenu-chiffre.bin | read HEXA
% echo $HEXA
16610e03c5c406a59cbeb6fa493c768d0a5f9db91bc46d3c4f660007816b2f6115ea2c6e9e2433ff4e92293678416dbe46826ee6c4ff23da:
< >
```

- L'application construit enfin une URL de requête pour le endpoint de KIF IdP, constituée de l'URL du endpoint dans laquelle un paramètre est inclus, nommé `msg` et contenant la représentation textuelle du message chiffré.

Voici un exemple d'une telle URL :

```
https://fenyo.net/fc/msg=16610e03c5c406a59cbeb6fa493c768d0a5f9db91bc46d3c4f660007816b2f6115ea2c6e9e2433ff4e92293678416dbe46826ee6c4ff23da:
< >
```

- L'application redirige le navigateur de l'utilisateur vers cette URL de requête.

### Création d'une réponse à une requête

Lorsque KIF-IdP reçoit une requête d'authentification, il engage la cinématique d'authentification avec FranceConnect en s'appuyant sur MITREid Connect au travers de KIF-SP, récupère l'identité de l'utilisateur et construit une réponse à destination de l'application :

- le paramètre `msg`, représentation textuelle du message chiffré, est extrait de la requête de l'application,

Dans notre exemple, KIF-IdP retrouve donc la chaîne suivante :

```
16610e03c5c406a59cbeb6fa493c768d0a5f9db91bc46d3c4f660007816b2f6115ea2c6e9e2433ff4e92293678416dbe46826ee6c4ff23da:
```



- cette chaîne de caractères, qui représente en hexadécimal une chaîne d'octets, est transformée en chaîne d'octets,
- cette chaîne est déchiffrée avec AES-256-CBC, en utilisant la clé secrète et le vecteur d'initialisation partagés avec l'application, ce qui produit la représentation binaire d'un message en clair,
- cette représentation binaire est transformée en chaîne de caractères à l'aide du charset US-ASCII, ce qui produit le message en clair,

Dans notre exemple, KIF-IdP retrouve donc la chaîne suivante :

```
https://fenyo.net/fc/identite.cgi?nonce=2ff22cb9663990d009fd0dfe87d997c6&state=f894bb7061a7c2a2
```

- le message en clair est une URL de callback vers l'application, contenant notamment les paramètres `state` et `nonce`, KIF-IdP vérifie que le début de cette URL correspond au contenu du paramètre de configuration `net.fenyo.franceconnect.config.idp.redirecturi`, afin de s'assurer que le message provient bien de l'application,
- KIF-IdP enrichit l'identité au format JSON, récupérée par KIF-SP, les paramètres `state` et `nonce`, ce qui constitue le message en clair qui doit être adressé au serveur d'application,

Dans notre exemple, KIF-IdP a reçu de FranceConnect le message JSON suivant, sous forme compacte :

```
{
  "sub": "54f70a557d838bcd26abd22038126819299ef2048c01eab97a7a10545976ef98v1",
  "gender": "male", "birthdate": "1981-06-23",
  "birthcountry": "99100",
  "birthplace": "91272",
  "given_name": "Eric",
  "family_name": "Mercier",
  "email": "eric.mercier@france.fr",
  "address": {
    "formatted": "26 rue Desaix, 75015 Paris",
    "street_address": "26 rue Desaix",
    "locality": "Paris",
    "region": "Ile-de-France",
    "postal_code": "75015",
    "country": "France"
  }
}
```

KIF-IdP enrichit donc ce message JSON comme ceci :

```
{
  "sub": "54f70a557d838bcd26abd22038126819299ef2048c01eab97a7a10545976ef98v1",
  "gender": "male", "birthdate": "1981-06-23",
  "birthcountry": "99100",
  "birthplace": "91272",
  "given_name": "Eric",
  "family_name": "Mercier",
  "email": "eric.mercier@france.fr",
  "address": {
    "formatted": "26 rue Desaix, 75015 Paris",
    "street_address": "26 rue Desaix",
    "locality": "Paris",
    "region": "Ile-de-France",
    "postal_code": "75015",

```



```

    "country": "France"
  },
  "nonce": "2ff22cb9663990d009fd0dfe87d997c6",
  "state": "f894bb7061a7c2a2"
}

```

- le message en clair est transformé dans sa représentation binaire,
- la représentation binaire est chiffrée avec AES-256-CBC, en utilisant la clé secrète et le vecteur d'initialisation partagés avec l'application, ce qui produit le message chiffré,
- le message chiffré est converti dans sa représentation textuelle,

Dans notre exemple, cela correspond à la représentation suivante :

```
b74f31907bb2d9be0ab2750c29dc4839061af809ada6217b237690a577f96f76d3f0f8633b28c8125aa89225b47930929e1e406a09ab6488614c3
```

- l'URL de callback vers l'application est enrichie d'un paramètre `info` contenant la représentation textuelle du message chiffré,

Dans notre exemple, cela correspond à l'URL suivante :

```
https://fenyo.net/fc/identite.cgi?nonce=2ff22cb9663990d009fd0dfe87d997c6&state=f894bb7061a7c2a2&info=b74f31907bb2d9be0ab2750c29dc4839061af809ada6217b237690a577f96f76d3f0f8633b28c8125aa89225b47930929e1e406a09ab6488614c3
```

- KIF-IdP redirige alors le navigateur de l'utilisateur vers cette URL de callback.



On peut remarquer que les paramètres `nonce` et `state` sont présents dans l'URL de callback puisqu'ils y étaient déjà présents dans le message chiffré de la requête de l'application. **Néanmoins, l'application doit ignorer ces paramètres présents de manière non chiffrée dans l'URL de callback, et uniquement se baser sur leur valeur dans le contenu JSON chiffré au sein du paramètre `info`, car rien ne permet de s'assurer que l'URL de callback n'a pas été forgée. C'est pour couvrir ce risque de sécurité en s'assurant que leur valeur provient bien de KIF-IdP que ces paramètres ont été rajoutés par ce dernier dans le message JSON.**

## Traitement d'une réponse

Lorsque l'application reçoit une réponse à une requête d'authentification, cette réponse est invoquée sur l'URL de callback initialement fournie dans la requête.

L'application traite alors cette réponse comme suit :

- Le paramètre `info` est extrait de la réponse et constitue la représentation textuelle du message de réponse chiffré.

Dans notre exemple, l'application récupère la chaîne suivante, produite par KIF-IdP :

```
b74f31907bb2d9be0ab2750c29dc4839061af809ada6217b237690a577f96f76d3f0f8633b28c8125aa89225b47930929e1e406a09ab6488614c3
```

- Cette représentation textuelle d'un message chiffré est convertie en chaîne d'octets.

Voici un exemple de conversion avec Perl5 :

```
% HEXA=b74f31907bb2d9be0ab2750c29dc4839061af809ada6217b237690a577f96f76d3f0f8633b28c8125aa89225b47930929e1e406a09ab64
% echo -n $HEXA | perl -pe 's/([0-9a-f]{2})/chr hex $1/gie' > contenu-chiffre.bin
%
```

- La chaîne d'octets est déchiffrée avec AES-256-CBC, en utilisant la clé secrète et le vecteur d'initialisation partagés avec KIF-IdP, ce qui produit la représentation binaire d'un message en clair.

Pour déchiffrer cette chaîne, on peut utiliser openssl :

```
% openssl aes-256-cbc -d -K $KEY -iv $IV < contenu-chiffre.bin > reponse.bin
%
```

- Cette représentation binaire est transformée en chaîne de caractères à l'aide du charset UTF-8, ce qui produit le message en clair.

Dans notre exemple, pour afficher cette chaîne, il suffit d'utiliser `/bin/cat` en ayant au préalable vérifié que la locale du terminal utilisé est correctement positionnée (UTF-8), afin que les éventuels accents dans l'état civil de l'utilisateur soient correctement affichés. Si on utilise `xterm`, on peut le lancer avec le paramètre `-en UTF-8` afin qu'il affiche correctement les caractères UTF-8. Dans un shell sous `xterm` lancé de cette manière, on se contente d'invoquer `/bin/cat` comme ceci :

```
% cat reponse.bin ; echo
{"sub": "54f70a557d838bcd26abd22038126819299ef2048c01eab97a7a10545976ef98v1", "gender": "male", "birthdate": "1981-06-23",
%

```



- Le message en clair est au format JSON, incluant l'identité de l'utilisateur et le contenu des paramètres `state` et `nonce` fournis dans la requête initiale.

## Vérifications de sécurité

Avant d'utiliser l'identité de l'utilisateur dans la réponse fournie par KIF-IdP, l'application doit réaliser deux vérifications de sécurité afin de s'assurer de l'intégrité des informations reçues ainsi que de leur provenance :

- l'application vérifie que le paramètre `state` contenu dans le message JSON correspond bien à la session utilisateur en cours, afin d'éviter les attaques par saut de session,
- l'application vérifie que le paramètre `nonce` contenu dans le message JSON correspond bien à celui fourni dans la requête d'authentification initiale, afin d'éviter les attaques par rejeu.

**Si l'une ou l'autre de ces vérifications s'avère négative, l'utilisateur doit être renvoyé vers une page d'erreur d'authentification.**

On peut noter que la vérification par l'application de la présence de paramètres connus dans la réponse chiffrée reçue correspond à une opération de vérification de signature. En effet, pour produire un message chiffré contenant ces valeurs imposées, il faut connaître la clé secrète partagée. L'application récupère donc ici la garantie que le message a été produit par KIF-IdP, sans avoir eu à implémenter un quelconque mécanisme de signature numérique.

## Sources de l'application exemple

L'application de démonstration est constituée de trois scripts, dont voici les contenus :

- script de la page d'accueil (<https://fenyo.net/fc/index.cgi>) :

```
#!/bin/zsh
KEY=a6a7ee7abe681c9c4cede8e3366a9ded96b92668ea5e26a31a4b0856341ed224
IV=87b7225d16ea2ae1f41d0b13fdce9bba
echo Content-type: text/html
echo Cache-Control: no-store
echo Pragma: no-cache
echo
echo
echo "<html><head><meta http-equiv='Content-Type' content='text/html; charset=UTF-8'><title>Demo France Connect</titl
openssl rand -hex 16 | head -c 16 | read STATE
openssl rand -hex 16 | head -c 32 | read NONCE
echo -n "https://fenyo.net/fc/identite.cgi?nonce=$NONCE&state=$STATE" | openssl aes-256-cbc -K $KEY -iv $IV | hexdump
echo "cliquez sur ce lien pour vous authentifier puis revenir à ce service :<br/>"
echo "<a href='http://127.0.0.1/idp?msg=$HEX'>http://127.0.0.1/idp?msg=$HEX</a></body></html>"

```



- script de la page protégée (<https://fenyo.net/fc/identite.cgi>) :

```
#!/bin/zsh
KEY=a6a7ee7abe681c9c4cede8e3366a9ded96b92668ea5e26a31a4b0856341ed224
IV=87b7225d16ea2ae1f41d0b13fdce9bba
echo Content-type: text/html
echo Cache-Control: no-store
echo Pragma: no-cache
echo
echo
echo "<html><head><meta http-equiv='Content-Type' content='text/html; charset=UTF-8'><title>Demo France Connect</t
echo "$QUERY_STRING" | sed 's/.info=\([a-z0-9]*\).*\/1/' | perl -pe 's/([0-9a-f]{2})/chr hex $1/gie' | openssl ae
if echo $IDENT | grep -v state > /dev/null ; then echo "Erreur d'authentification" ; exit 1 ; fi
echo "$IDENT" | jq -r '.given_name, family_name' | xargs echo | read NAME

```



```
echo '<script src="https://fcp.integ01.dev-franceconnect.fr/js/franceconnect.js"></script>'
echo '<div style="color: #000000; background-color: #000ccc" id="fconnect-profile" data-fc-logout-url="logout.cgi"><t
echo "<a href="#">$NAME</a><br/> </div>"
echo "Vous êtes authentifié : $IDENT"
echo "<p>IMPORTANT : pour assurer la protection anti-rejeu et contre le saut de session, le programmeur doit vérifie
echo "</body></html>"
```

- script de déconnexion (<https://fenyo.net/fc/logout.cgi>) :

```
#!/bin/zsh
echo Content-type: text/html
echo Location: http://127.0.0.1/j_spring_security_logout
echo
echo
echo '<a href="http://127.0.0.1/j_spring_security_logout">cliquez ici</a>'
```

## Configuration

KIF-IdP s'appuie sur KIF-SP pour la relation avec FranceConnect. Sa configuration est présentée dans la [section correspondante de KIF-SP](#).

## Traces

En complément des traces de KIF-SP, KIF-IdP produit les traces suivantes, au même format que celles de KIF-IdP :

- accès à `/idp` : cette trace est produite lorsque KIF-IdP a récupéré l'identité de l'utilisateur pour le compte de l'application
- accès à `/idp: redirection [URL de redirection]` : cette trace indique l'URL de redirection de l'application vers laquelle l'utilisateur authentifié est redirigé, au moment où cette redirection se produit
- accès à `/idp: requête chiffrée [requête chiffrée]` : cette trace permet de conserver le message chiffré par l'application
- accès à `/idp: requête déchiffrée [requête déchiffrée]` : cette trace permet de conserver le message envoyé par l'application, après qu'il ait été correctement déchiffré
- accès à `/idp: exception` : cette trace indique qu'une erreur de chiffrement/déchiffrement ou de conversion d'un message de l'application s'est produite (l'utilisateur est redirigé vers une page signalant une erreur d'authentification)
- accès à `/idp: renvoi vers la page d'erreur d'authentification (null nonce)` : cette trace indique que la requête ne contient pas de paramètre `nonce` (l'utilisateur est redirigé vers une page signalant une erreur d'authentification)
- accès à `/idp: renvoi vers la page d'erreur d'authentification (null state)` : cette trace indique que la requête contient un paramètre `nonce` mais pas de paramètre `state` (l'utilisateur est redirigé vers une page signalant une erreur d'authentification)
- accès à `/idp: renvoi vers la page d'erreur d'authentification (url de callback invalide)` : cette trace indique que la requête ne commence pas par le début de l'URL de callback qui est configuré dans le paramètre `net.fenyo.franceconnect.config.idp.redirecturi`, ce qui prouve que cette requête ne provient pas de l'application (l'utilisateur est redirigé vers une page signalant une erreur d'authentification)

## Déploiement

KIF-IdP étant intégré à KIF-SP, le déploiement est réalisé de la même manière que celui décrit dans la [section correspondante de KIF-SP](#).

## Support

Un support en mode *best-effort* est fourni en contactant l'adresse suivante :

[support-fc@fenyo.net](mailto:support-fc@fenyo.net)

written with StackEdit - Support StackEdit



[StackEdit](#) is a full-featured, open-source Markdown editor based on PageDown, the Markdown library used by Stack Overflow and the other Stack Exchange sites.

