[archivé]

Documentation en partie obsolète, merci de vous référer à ESUP-SGC-Client et édition des cartes

esup-sgc-client correspond au client esup-sgc/esup-nfc-tag permettant

- après impression de la carte l'enrolement de la carte physique électronique dans esup-sgc (via lecture du CSN)
- l'encodage de la carte, grâce à esup-nfc-tag et éventuellement grâce à esup-cnous-client (pour les cartes non pré-encodées crous/izly)

2 clients esup-sgc-client sont proposés depuis le dépot git https://github.com/EsupPortail/esup-sgc-client :

- https://github.com/EsupPortail/esup-sgc-client/releases/tag/esupsgcclient-v2.0 (Encodage manuel)
- https://github.com/EsupPortail/esup-sgc-client/releases/tag/esupsgcclient-r2d2-v2.0 (Encodage via le robot ZXP3)

esup-sgc-client demande une connexion shibboleth via une webview intégrée (identique à l'application mobile) et utilise un localstorage pour stocker les informations nécessaires.

L'application doit pouvoir écrire un fichier localstorage qui se situe :

- sous windows, dans le home de l'utilisateur dans /AppData/Local/EsupSgcClient/
- sous linux dans le répertoire courant depuis lequel on lance la commande Java

Pour que le client Esup-Sgc-Client voit l'application d'écriture SGC configurée dans Esup-NFC-TAG-Server, il faut que l'option "Visible" soit cochée dans l'interface d'Esup-NFC-TAG-Server.

ESUP-SGC-CLIENT

Fonctionnalités

1. L'application lit le QR code imprimé sur la carte à encoder qui correspond à l'identifiant du futur propriétaire de la carte.
2. Demande la sélection dans esup-sgc de l'individu à encoder
3. L'application récupéré les commandes à exécuter sur la carte via esup-nfc-tag-server
4. Validation de l'encodage et activation de la carte
5. Éventuellement encodage de l'application CROUS (voir https://www.esup-portail.org/wiki/display/SGC/FAQ#FAQ-Peut-onencoderl'applicationCROUSquandonutilisedescartesvierges?)

Environnement

Logiciel

- L'application fonctionne sous java avec JavaFX.

- L'application fonctionne sur Linux ou Windows 10 64bits (l'encodage CROUS necessite Windows 10 64bits)
- pour l'encodage de l'application CROUS le pilote OMNIKEY CardMan 6121 (pilote) (voir ESUP-CNOUS-CLIENT)

Materiel

L'application nécessite :

- une webcam gérant la résolution VGA (640x480)
- un lecteur de carte compatible PC/SC
- pour l'encodage de l'application CROUS il faut connecter la clé SAM OMNIKEY CardMan 6121 avec sa carte sim

La webcam est placée pour filmer le lecteur de carte (procéder à la mise au point si besoin). Lorsqu'une carte est posée sur le lecteur de carte, la webcam détecte le QR code et la procédure d'encodage démarre

Documentation de mise en œuvre ESUP-SGC / ESUP-NFC-TAG#SGC/ESUP-NFC-TAG-Installationmaterielle

Installation

Le plus rapide est de récupérer le jar esup-sgc-client et/ou l'installateur windows depuis https://esup-sgc-client-web-installer.univ-rouen.fr/

Vous pouvez le packager/builder vous même également cependant, cf ci-dessous.

Sources :

https://github.com/EsupPortail/esup-sgc-client.git

```
git clone https://github.com/EsupPortail/esup-sgc-client.git
git checkout -b esupsgcclient-univ-ville esupsgcclient-v2.3
```

Paramétrage

Il faut modifier le fichier src/main/resources/esupsgcclient.properties pour y mettre vos adresses esup-nfc-tag-server et esup-sgc et activer ou non l'encodage CNOUS

```
esupSgcUrl = https://esup-sgc.univ-ville.fr
esupNfcTagServerUrl = https://esup-nfc-tag.univ-ville.fr
encodeCnous = false
localStorageDir = /AppData/Local/EsupSgcClient/
```

Compilation esup-sgc-client

Dans le dossier esup-sgc-client executer :

```
mvn clean package
```

Pour disposer du jar au sein du site ESUP-SGC :

Copier le fichier esup-sgc-client-final.jar (répertoire target) en le renommant **esupsgcclient-shib.jar** dans la webapp esup-sgc ou dans vos sources sous src/main/webapp/ avant de compiler esup-sgc

pour le client zxp3 le fichier (esupsgcclient-**-jar-with-dependencies.jar) doit être renommé **esupsgcclient-r2d2-shib.jar.**

Lancement

Si vous utilisez l'installateur windows, vous n'avez qu'à lancer le client depuis le raccourci créé par celui-ci, l'installateur embarque directement une distribution d'un openjdk et openjfx proposés par la communauté Zulu !

Vous pouvez aussi utilise openJDK fourni depuis https://jdk.java.net et openJFX fourni depuis https://gluonhq.com/products/javafx/ (voir également https://openjfx.io/openjfx-docs/) qui sont des produits opensource et gratuits également.

esup-sgc-client est compatible avec les JDK 9 et supérieurs. Depuis juin 2022, l'installateur windows embarque les versions JDK/JFX 18 packagées par Zulu (les dernières en date).

Suivant l'intégration de JAVAFX, la commande à lancer est la suivante :

```
java.exe --module-path %PATH_TO_FX% --add-modules javafx.controls,javafx.fxml,javafx.base,javafx.media,javafx.
graphics,javafx.swing,javafx.web -Dcom.sun.webkit.useHTTP2Loader=false -jar esupsgcclient-shib.jar
```

L'option "-Dcom.sun.webkit.useHTTP2Loader=false" donnée ici à titre indicatif permet de forcer l'usage de HTTP 1 et peut permettre d'éviter d'éventuels problèmes avec des serveurs web (apache) un peu anciens.

Précisions

Les installations d'openJDK et OpenJFX correspondent à dézipper leurs archives dans un répertoire donné.

Sur un poste windows, on peut par exemple les dézipper dans un répertoire C:\esup-sgc-client, on aura ainsi

- C:\esup-sgc-client\jdk-12.0.1 pour le JDK
- C:\esup-sgc-client\javafx-sdk-12.0.1 pour le JavaFX

En plaçant le JAR d'esup-sg-client (esupsgcclient-shib.jar) dans ce même répertoire C:\esup-sgc-client on peut ainsi finalement exécuter le JAR ainsi :

```
C:\esup-sgc-client\jdk-12.0.1\bin\java.exe --module-path C:\esup-sgc-client\javafx-sdk-12.0.1\lib --add-modules
javafx.controls,javafx.fxml,javafx.base,javafx.media,javafx.graphics,javafx.swing,javafx.web -jar C:\esup-sgc-
client\esupsgcclient-shib.jar
```

Si vous utilisez cette commande au travers d'un raccourci windows (pour faciliter son usage c'est une bonne idée, et c'est ce que propose l'installateur windows que vous pouvez générer depuis https://esup-sgc-client-web-installer.univ-rouen.fr), n'oubliez pas également de préciser le répertoire d'exécution comme étant également C:\esup-sgc-client.

A l'exécution ce client esup-sgc-client esupsgcclient-shib.jar écrit en effet 2 fichiers dans le répertoire courant :

1. un fichier de logs pour éventuellement tracer les actions/infos/erreurs
2. un fichier contenant un jeton d'authentification permettant à l'utilisateur d'éviter de devoir resssisir ses identifiants à chaque lancement de l'application.

ESUP-SGC-CLIENT-ZXP3

Esup-sgc-client-zxp3 est l'application permettant d'encoder les cartes Mifare Desfire dans le cadre du Système de gestion de carte Esup-sgc. Elle est identique à Esup-sgc-client mais elle utilise une imprimante Zebra ZXP3 pour automatiser l'encodage.

Environnement

Logiciel

- OS Windows 10 64bits (L'application devrait pouvoir tourner sous Linux si l'encodeur SDI010 est bien reconnu)
- L'application est prévue pour tourner sur du **java 8 avec javafx** (le SDK de Zebra requiert un JDK8 et n'est donc pas compatible avec un JDK11 par exemple). La version du JDK 8 d'Oracle embarque javafx et fonctionne, mais suite au changement de license côté Oracle, il faudrait maintenant s'acquitter d'une license pour un usage en production. Aussi le mieux ici est de prendre la version d'**OpenJdk avec OpenJFX de com munauté Zulu** Nous en gardons une copie ici : zulu8.44.0.13-ca-fx-jdk8.0.242-win_x64.zip
- Le pilote Zebra ZXP3
- Le pilote PCSC SDI010
- Le SDK Zebra LinkOs
- Pour permettre l'encodage CNOUS il faut utiliser une machine windows 64bits pour lancer le client et il faut avoir installé l'application CNOUS Espu-sgc-cnous (voir ESUP-CNOUS-CLIENT)

Materiel

L'application nécessite :

- une webcam gérant la resolution VGA (640x480)
- une imprimante Zebra ZXP3 avec encodeur SDI010
- Pour l'encodage de l'application CNOUS il faut connecter la clé SAM OMNIKEY CardMan 6121 avec sa carte sim

La webcam est placée dans l'imprimante (qui reste ouverte) pour filmer le lecteur de carte. Il faut donc placer quelque chose dans le capteur de fermeture du couvercle. voir : https://www.esup-portail.org/wiki/pages/viewpage.action?pageId=613384398

Reglage offset lecteur sans contact à 0

Via les outils du driver (sous windows "Propiétés de l'imprimante > Device Settings > Tools > Command to send to printer") lancer la commande : `+OS 0`

réponse : `0 <ACK>`

Installation

Sources : https://github.com/EsupPortail/esup-sgc-client/tree/univ-rouen-robot-zxp3

```
git clone https://github.com/EsupPortail/esup-sgc-client.git
git checkout -b esupsgcclient-r2d2-v2.0 esupsgcclient-r2d2-v2.0
```

Compilation Maven

Installation des dépendances ZSDK_API et ZSDK_CARD_API

Pour communiquer avec la Zebra ZXP3 esup-sgc-client-zxp3 utilise le SDK Zebra. Pour fonctionner il est installé en tant que depot maven local.

Le sdk se récupère à cette adresse : https://www.zebra.com/fr/fr/products/software/barcode-printers/link-os/link-os-sdk.html

Après l'installation du mpsdk-installer, vous pouvez ainsi récupérer et copier les deux jar présents dans le dossier link_os_sdk/PC-Card/v2.12.3968/lib :

- ZSDK_API.jar, renomé ZSDK_API-2.12.3968.jar, dans le dossier src/lib/com/zebra/sdk/comm/ZSDK_API/2.12.3968/
- ZSDK_CARD_API.jar, renomé ZSDK_CARD_API-2.12.3968.jar, dans le dossier src/lib/com/zebra/sdk/common/card/ZSDK_CARD_API/2.12.3968/

Pour la compilation avec maven, copier tout le dosier com situé dans src/lib dans le dossier ~/.m2/repository

Compilation maven

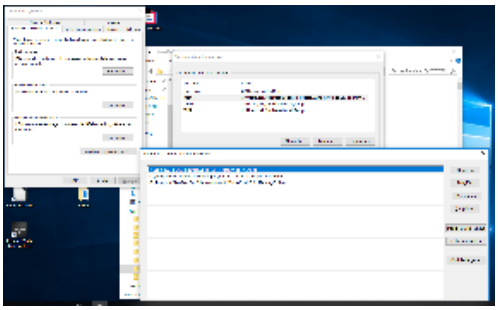
Depuis le répertoire source on lance :

```
mvn package
```

Vous pouvez alors renommer target/esupsgcclient-r2d2-2.1-SNAPSHOT-jar-with-dependencies.jar en esupsgcclient-r2d2-shib.jar.

Lancement depuis un poste windows

Sur le poste client il vous faut également indiquer dans le path le répertoire link_os_sdk/PC-Card/v2.12.3968/lib en plus du répertoire bin de votre JDK+JFX Zulu - voir copié d'écran :



Avec JDK+JFX Zulu, et en valuant ainsi votre PATH, vous n'avez plus qu'à lancer le jar ainsi :

```
java -jar esupsgcclient-r2d2-shib.jar
```

Note : documentation du SDK : http://techdocs.zebra.com/link-os/2-12/