Installation / configuration des anciens clients ESUP-SGC-CLIENT-V1.0

Page de documentation obsolète, merci d'utiliser ESUP-SGC-CLIENT-V2.0 : ESUP-SGC-Client et édition des cartes

Documentation [archivé]

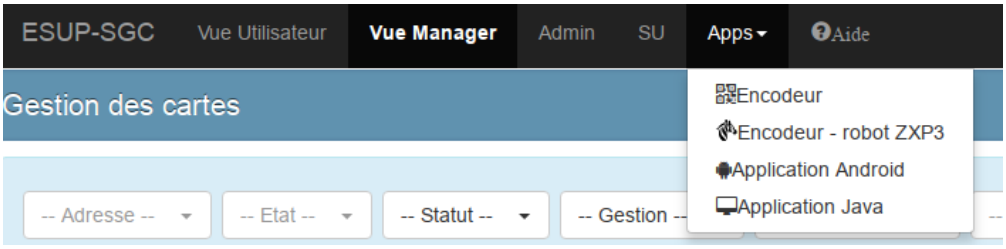
2 encodeurs sont disponibles dans ESUP-SGC, ils se lancent via JavaWebStart (JWS) depuis un fichier JNLP téléchargeable depuis le serveur esup-sgc. Pour que cela fonctionne, et vis à vis des contrôles de sécurité Java, les jar pointés par les fichiers JNLP doivent être signés ; voir à ce propos la page de documentation Signature de code / signature d'un JAR.

- esup-sgc-client : encodeur livré sous forme de jar dans les sources d'esup-sgc ; le jar livré est signé par Vincent Bonamy de l'Université de Rouen ; vous pouvez le recompiler et le signer vous-même, mais ce n'est pas obligatoire.
- esup-sgc-client-zxp3 : encodeur 'de masse' utilisant unz zxp3, il dépend de jar et dll Zebra que l'on ne peut s'autoriser à distribuer ; si vous souhaitez utiliser cet encodeur, vous devez le compiler vous-même (et signer le jar par vous-même).

ESUP-SGC-CLIENT

Esup-sgc-client est l'application permettant d'encoder les cartes Mifare Desfire EV1 dans le cadre du Système de gestion de carte ESUP-SGC. Le client s'appuie aussi sur la plateforme ESUP-NFC-TAG qui calcule les commandes (APDU) à transmettre à la carte.

L'application est packagée sous la forme d'un jar comprenant les dépendances : esupsgcclient-1.0-SNAPSHOT-jar-with-dependencies.jar. Le jar destinée à être lancé en mode java web start (jws) depuis l'application web Esup-sgc



Lors du lancement de l'application via ESUP-SGC -> Apps -> Encodeur (ou Encodeur - robot ZXP3) ESUP-SGC crée automatiquement un périphérique dans ESUP-NFC-TAG.

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 est prévue pour tourner sur du java 8 (+ java FX). Elle est lancée en JWS il faudra donc autoriser l'application qui va demander tous les droits sur la machine cliente.
- 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

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

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

Compilation esup-sgc-client

Dans le dossier esup-sgc-client executer :

```
mvn clean package
```

A la fin de la compilation le jar esupsgcclient-1.0-SNAPSHOT-jar-with-dependencies.jar va être signé (l'application doit être signée car elle demande une élévation de droits au lancement). Le keystore pour la signature est déclaré dans le pom.xml :

```
<configuration>
        <keystore>src/etc/keystore.jks</keystore>
        <alias>server</alias>
        <storepass>leocarte</storepass>
        <keypass>leocarte</keypass>
</configuration>
```

Copier le fichier esupsgcclient-1.0-SNAPSHOT-jar-with-dependencies.jar à la racine de votre webapp esup-sgc ou dans vos sources sous src/main /webapp/ avant de compiler esup-sgc

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. Elle est lancée en JWS il faudra donc autoriser l'application qui va demander tous les droits sur la machine cliente.
- 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 univ-rouen-robot-zxp3
```

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 il faut 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

Il faut, de plus, copier les dll ZebraNativeUsbAdapter_32.dll et ZebraNativeUsbAdapter_64.dll dans c:\Windows\System32 et c:\Windows\SysWOW64 du poste client

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

Compilation esup-sgc-client-zxp3

Dans le dossier esup-sgc-client executer :

```
mvn clean package
```

A la fin de la compilation le jar esupsgcclient-r2d2-1.0-SNAPSHOT-jar-with-dependencies.jar va être signé (l'application doit être signée car elle demande une élévation de droits au lancement) Le keystore pour la signature est déclaré dans le pom.xml :

```
<configuration>
        <keystore>src/etc/keystore.jks</keystore>
        <alias>server</alias>
        <storepass>leocarte</storepass>
        <keypass>leocarte</keypass>
</configuration>
```

Copier le fichier esupsgcclient-r2d2-1.0-SNAPSHOT-jar-with-dependencies.jar vers esupsgcclient-r2d2.jar à la racine de votre webapp esup-sgc ou dans vos sources sous src/main/webapp/ avant de compiler esup-sgc

ESUP-CNOUS-CLIENT

voir la FAQ : FAQ

L'application esup-cnous-client permet la lecture et l'encodage de l'application CNOUS sur une carte préalablement configurée (application desfire F58540 déjà présente)

Fonctionalitées

- CreationCarteCrous.exe ou CreationCarteCrous.exe -h : affiche l'aide
- CreationCarteCrous.exe -t : contrôle de l'application. Retourne true si tout est OK
- CreationCarteCrous.exe -l : lit l'application CNOUS de la carte présente sur le lecteur
- CreationCarteCrous.exe -e XXXXXXXXXXX : encode l'application CNOUS avec le numéro passé en paramètre (le numéro doit comporter 15 caractères)

Environnement

Matériel

- un lecteur de carte compatible PC/SC
- une clé SAM OMNIKEY CardMan 6121 (l'application utilise automatiquement la clé de type 6121 comme clé SAM et l'autre lecteur de carte comme encodeur)

Logiciel

- un environnement windows 64bits
- le framework .net 2 minimum

Installation

Pour un fonctionnement avec le client esup-sgc-client, l'exécutable doit être copié dans le dossier c:\cnousApi du poste sur lequel sera lancé l'encodage. De plus, l'application nécessite la présence de 4 librairies dans ce même dossier:

- cnous_fournisseur_carte.dll (à demander auprès du CNOUS)
- libeay32.dll (openssl)
- libssl32.dll (openssl)
- pcsc_desfire.dll (springcard)

la version de la dll cnous doit etre x64

Sources : https://github.com/EsupPortail/esup-crous-client

```
git clone https://github.com/EsupPortail/esup-crous-client.git
```

Compilation

Pour compiler la solution il suffit de télécharger la dernière version de Visual Studio Community (https://www.visualstudio.com/fr/downloads/) ainsi que le framework .net version > 2