Configurations ESUP-SGC et ESUP-NFC-TAG-SERVER

Introduction

Comme dit précédemment, le SGC nécessite une instance d'ESUP-NFC-TAG-SERVER pour fonctionner. Dans ce document on présente donc également les modifications de configuration qu'il faudra effectué au niveau d'ESUP-NFC-TAG-SERVER.

Versionning de la configuration

D'une manière générale il est conseillé de faite des commits GIT locaux pour sauvegarder vos modifications de configuration.

Par exemple, lors que vous avez fini la configuration du fichier "src/main/resources/META-INF/spring/applicationContext-custom.xml" vous pouvez exécuter les commandes suivantes depuis la racine de vos sources (ex : /opt/esup-nfc-tag):

```
git add src/main/resources/META-INF/spring/applicationContext-custom.xml
git commit -m "config prod univ-ville-fr"
```

La modification du fichier applicationContext-custom.xml sera versionnée ce qui permet de conserver les traces de toutes les modifications (procédure à faire à chaque modification). De plus lors d'une mise à jour d'ESUP-SGC (git pull) la configuration ne sera pas écrasée.

ESUP-SGC

La configuration se fait via les fichiers suivants :

- src/main/resources/META-INF/spring/database.properties
- src/main/resources/META-INF/persistence.xml
- src/main/resources/META-INF/spring/applicationContext-services.xml
- src/main/resources/META-INF/spring/applicationContext-crous.xml
- src/main/resources/META-INF/spring/applicationContext-paybox.xml
- src/main/resources/META-INF/spring/applicationContext-access-control.xml
- src/main/resources/META-INF/spring/applicationContext-security.xml

Paramétrage de la base de données (database.properties et persistence.xml)

Les paramètres de connexion à la base de données sont indiqués dans le **database.properties**. Par exemple:

```
database.driverClassName=org.postgresql.Driver
database.url=jdbc\:postgresql\://localhost\:5432/esupsgc
database.username=esupsgc
database.password=esup
```

applicationContext-services.xml

UserInfo :

La première partie du fichier comporte les « UserInfoService » ainsi que les « SpelUserInfoService » . Ces deux entités ont pour but de peupler les informations des demandeurs de carte à l'aide des différentes sources de données présentent dans le SI de l'établissement (ou dans les SI des établissements partneaires - cas d'une authentification shibboleth extérieure, ou encore de l'usage d'un 'meta-annuaire' multi-établissements).

Trois methodes sont implémentées dans le SGC :

- ShibUserInfoService (recupération des données via le context Shibboleth)
- LdapUserInfoService
- SqlUserInfoService

Les spelUserInfoService sont des règles qui s'appliquent ensuite pour calculer certains attributs de l'utilisateur.

La recherche dans les différentes sources ainsi que le calcul des règles se fait de manière séquentielle dans l'ordre défini par la propriété « p:order ». A chaque étape, quand une donnée est trouvée, elle remplace celle trouvée par l'étape précédente.

C'est via les 'userInfos' que la synchronisation des données utilisateurs depuis le Système d'Information se fait. Les champs 'userInfos' sont de tout type : nom, prénom, date de naissance, indice, numéro étudiant, numéro personnel, email, libellés à écrire sur la carte, thème de la carte, etc.

On privilégie l'usage de champs utilisateurs 'standardisés'. En ce sens on privilégie notamment l'usage de champs supann depuis shib/ldap.

On note que l'usage de shibboleth permet de récupérer des informations sur un individu extérieur au SI sans connexion directe d'Esup-SGC au SI de cet individu. Le fonctionnement de shibboleth fait cependant que ces informations ne pourront pas être mises à jour régulièrement sans connexion régulière de l'individu au SGC. Aussi la récupération des informations également depuis le LDAP au moins pour les utilisateurs internes au SI est à implémenter (en place ou même en plus de shibboleth).

Nom du champ esup-sgc	Usage	Obligatoire	Source à 'privilégier' / Format
eppn	Identifiant métier : eduPersonPrincipalName	Oui	eduPersonPrincipalName - obligatoire dans toutes les sources - car permet de faire le lien -> shib, ldap et sql
email	Envoi d'email d'information lors de l'évolution de la carte ; ticket paybox, ... et aussi pour crous/izly obligatoire	Oui	shib/ldap - champ mail
eduPersonPrimaryAffiliation	Catégorisation population – moteur de recherche	Non	shib/ldap - champ eduPersonPrimaryAffiliation
supannEtuId	moteur de recherche	Non	shib/ldap - champ supannEtuId
supannEmpId	moteur de recherche	Non	shib/ldap - champ supannEmpId
supannCodeINE	affichage / construction identifiant ESCR / identfiant crous/izly pour les étudiants obligatoire dans CROUS/IZLY et ESCR	Oui	shib/ldap - champ supannCodeINE
supannEntiteAffectationPrincipale	moteur de recherche	Non	shib/ldap - champ supannEntiteAffectationPrincipale
firstname	Affichage / moteur de recherche	Oui	shib/ldap - champ givenname
name	Affichage / moteur de recherche	Oui	shib/ldap - champ sn
schacDateOfBirth	Date de naissance - obligatoire dans les contrôles d'accès. Format : yyyyMMdd	Oui	shib/ldap - champ schacDateOfBirth
schacExpiryDate	Date de fin de droits – les cartes de l'individu sont marquées comme caduques cette date passée. Format UTC : yyyyMMddHHmmss'Z' Cette date est également envoyée aux services qui ont besoin d'une date de fin (expiration) telles que les contrôles d'accès, ESCR, le CROUS (dueDate).	Oui	shib/ldap - champ schacExpiryDate
referenceStatut	Population crous (psg, etd, prs, hbg, fct, fpa, stg) - permet de calculer le tarif et société crous depuis le fichier ESIST.xml	Oui	sql
indice	Indice du personnel - permet de calculer le tarif et société crous depuis le fichier ESIST.xml	Non	sql

secondaryId	Identifiant secondaire quelconque – affichage / moteur de recherche / web service **spécificité COMUE NU** : doit correspondre au leocode	Non	...
institute	Établissement, affichage ...	Non	règle d'un spelUserInfoService : nom (libellé) de l'établissement
supannEtablissement	Code RNE Établissement - permet de calculer le tarif et société crous depuis le fichier ESIST.xml Depuis la 1.1.1, ce code RNE ainsi récupéré est envoyé dans l'API CROUS en tant que rneOrgCode	Oui	shib/ldap - champ supannEtablissement code UAI / RNE : "{UAI}0761904G" pour désigner l'université de Rouen
address	Affichage / moteur de recherche Est affiché par défault dans le champ "Courrier Interne" dans le bloc "Adresse" du formulaire de demande de carte.	Non	peut correspondre à supannEntiteAffectationPrincipale ou champs sql spécifique
externalAddress	Affichage / moteur de recherche Est affiché par défault dans le champ "Domicile" dans le bloc "Adresse" du formulaire de demande de carte.	Non	
recto1	Libellé sur recto	Non	ldap, sql, ou/et calculé via des règles d'un spelUserInfoService
recto2	Libellé sur recto	Non	ldap, sql, ou/et calculé via des règles d'un spelUserInfoService
recto3	Libellé sur recto	Non	ldap, sql, ou/et calculé via des règles d'un spelUserInfoService
recto4	Libellé sur recto	Non	ldap, sql, ou/et calculé via des règles d'un spelUserInfoService
recto5	Libellé sur recto	Non	ldap, sql, ou/et calculé via des règles d'un spelUserInfoService
verso1	Libellé sur verso (dématérialisé)	Non	ldap, sql, ou/et calculé via des règles d'un spelUserInfoService
verso2	Libellé sur verso (dématérialisé)	Non	ldap, sql, ou/et calculé via des règles d'un spelUserInfoService
verso3	Libellé sur verso (dématérialisé)	Non	ldap, sql, ou/et calculé via des règles d'un spelUserInfoService
verso4	Libellé sur verso (dématérialisé)	Non	ldap, sql, ou/et calculé via des règles d'un spelUserInfoService
verso5	Libellé sur verso (dématérialisé)	Non	ldap, sql, ou/et calculé via des règles d'un spelUserInfoService
freeField1	Champ libre - peut servir à la recherche/affichage	Non	libre
freeField2	Champ libre - peut servir à la recherche/affichage	Non	libre
freeField3	Champ libre - peut servir à la recherche/affichage	Non	libre
supannRefId4ExternalCard	supannRefId donnant des numéros de cartes 'externes', cad non issus du SGC (et donc issus d'un autre SGC) Déprécié : merci d'utiliser csn4ExternalCard et control4ExternalCard	Non	shib/ldap - champ supannRefId - atend en dur {ISO15693}le-numero-csn {LEOCARTE:ACCESS-CONTROL}le-numero-access-control
csn4ExternalCard	csn des cartes 'externes', cad non issus du SGC (et donc issus d'un autre SGC)	Non	le-numero-csn (on pourra utiliser le champ supannRefId pour le faire transiter)
access-control4ExternalCard	numéro de carte de contrôle d'accès des cartes 'externes', cad non issus du SGC (et donc issus d'un autre SGC)	Non	le-numero-access-control (on pourra utiliser le champ supannRefId pour le faire transiter)

jpegPhoto4ExternalCard	jpegPhoto donnant la photo de cartes 'externes', cad non issus du SGC (et donc issus d'un autre SGC)	Non	shib/ldap - champ jpegPhoto sql : jpegPhoto correspondant à une photo jpeg exportée en base64
jpegPhoto	jpegPhoto donnant la photo par défaut pour un utilisateur : c'est cette photo qui est proposée par défaut lors de la première demande de carte ou via la demande de carte via l'API REST (sans spécifier une photo spécifique)	Non	shib/ldap - champ jpegPhoto sql : jpegPhoto correspondant à une photo jpeg exportée en base64
userType	**affichage** : onglet par userType **gestionnaire** par userType **spécificité COMUE NU** : ComueNuAccessControlCardIdService calcule un identifiant de carte à destination d'une application desfire pour contrôle d'accès en fonction de userType (P, E ou I).	Oui	règle d'un spelUserInfoService **Attention, le nombre maximal de caractères du userType est limité à 3** ; les libellés à donner dans les messages i18n sont libres quand à eux ; *vous pouvez modifier /ajouter ces libellés dans src/main/webapp/WEB-INF/i18n/sgc-messages.properties*
template	thème (template) de carte à utiliser	Oui	règle d'un spelUserInfoService - doit correspondre à une clef de template
editable	true ou false selon que l'utilisateur est 'éditable ou non'	Non	**remplace** l'usage de ROLE_USER_NO_EDITABLE (qui en tant que 'groupe' doit forcément correspondre à un groupe /champ ldap) si utilisé, il vous faut supprimer la référence à ROLE_USER_NO_EDITABLE dans sgcMappingGroupesRoles (et inversement)
requestFree	true ou false selon que l'utilisateur doit payer le renouvellement de la carte	Non	**remplace** l'usage de ROLE_USER_RENEWAL_PAYED (qui en tant que 'groupe' doit forcément correspondre à un groupe/champ ldap) si utilisé, il vous faut supprimer la référence à ROLE_USER_RENEWAL_PAYED dans sgcMappingGroupesRoles (et inversement)
blockUserMsg	Texte HTML affiché à l'utilisateur (vue utilisateur) en lieu et place du formulaire de demande de carte. Si vide (ou non spécifié dans la configuration), le formulaire de demande eest inchangé simplement.	Non	règle d'un spelUserInfoService - peut permettre de personnaliser le message disant à l'utilisateur qu'il ne peut pas demander de carte actuellement via eusp-sgc (car son inscription n'est pas en règle par exemple).
synchronize	Champ permettant d'indiquer que l'utilisateur (et ses cartes) doit être synchronisé depuis les userInfoServices. synchronize est à 'true' par défaut (fonctionnement usuel), si celui-ci est donné à false l'utilisateur et ses cartes restent figées dans esup-sgc : celà permet par exemple de conserver les anciennes cartes d'utilisateurs plus remontés par le SI (ldap/bd). A noter que même si synchronize est à false, le changement d'état de la carte en caduque en fonction de la date de fin enregistrée en base (schacExpiryDate) s'opère et provoque alors à cette date une mise à jour des informations utilisateurs/cartes sur les services tels que le contrôle d'accè, l'API CROUS, le LDAP ...	Non	règle d'un spelUserInfoService - dans la configuration donnée par défaut, on met synchronize à 'false' pour les utilisateurs n'ayant plus d'adresses mail de renseignées dans le Système d'Information (ldap/bd).

academic Level	Champ academicLevel permettant de transmettre le niveau d'étude de l'étudiant (en cours) à l'API ESCR (carte étudiante européenne). Ce niveau, si transmis, est notamment affiché derrière l'url correspondant au qrcode de la carte étudiante européenne (page internet précisant si l'individu est effectivement étudiant).	Non	Cf la doc de l'API ESCR : Current academic level of the student. Possibles values are : 6 - bachelor's degree. 7 - master's degree. 8 - doctorate.

Note supplémentaire sur le editable (et requestFree qui se comporte de la même manière) : après avoir utilisé un temps le userInfo editable (ou requestFree) et que vous voulez finalement revenir à l'usage de ROLE_USER_NO_EDITABLE (ou ROLE_USER_RENEWAL_PAYED),
il vous faut également supprimer les références au groupe *ROLE_USER_NO_EDITABLE (ou ROLE_USER_RENEWAL_PAYED) et remettre les editable à true (ou request_free à true) dans la base esupsgc pour réinitialiser les caculs :*
delete from roles where role='ROLE_USER_NO_EDITABLE';
update user_account set editable = true;
ou
delete from roles where role='ROLE_USER_RENEWAL_PAYED';

update user_account set request_free = true;

caducIfEmpty

Si vous souhaitez rendre caduques les cartes des utilisateurs qui ne sont plus remontés par le SI (ou qui n'ont plus le droit d'avoir de carte selon des règles internes, comme l'appartenance à un groupe ldap) alors même que la date de fin (schacExpiryDate dans esup-sgc) qui a été récupérée lorsque celle-ci était encore disponible reste postérieure à la date du jour, l'utilisation de caducIfEmpty peut être utile.

Dans applicationContext-services.xml vous pouvez mettre :

```
<bean id="userInfoService" class="org.esupportail.sgc.services.userinfos.UserInfoService">
  <property name="caducIfEmpty" value="caducIfEmpty"/>
</bean>
```

Par défaut la propriété caducIfEmpty étant valuée à ""

Vous pouvez alors manipuler un champ caducIfEmpty ainsi par exemple :

```
<bean id="NoCaduc4All" class="org.esupportail.sgc.services.userinfos.SpelUserInfoService" p:order="5">
  <property name="sgcParam2spelExp">
    <map>
      <entry key="caducIfEmpty" value="'Carte-OK'"/>
    </map>
  </property>
</bean>

<bean id="caducIfNoResourceLeocarte" class="org.esupportail.sgc.services.userinfos.SpelUserInfoService" p:
order="6">
  <property name="eppnFilter" value=".*@univ-ville\.fr"/>
  <property name="sgcParam2spelExp">
    <map>
      <entry key="caducIfEmpty" value="#userInfosInComputing['memberOf']!=null and #userInfosInComputing
['memberOf'].contains('cn=from.si.ressources.carte,ou=groups,dc=univ-ville,dc=fr')) ? 'Carte-OK' : ''"/>
    </map>
  </property>
</bean>
```

Avec une telle configuration, les utilisateurs univ-ville verront leurs cartes devenir caduques si le memberOf (groupes de l'utilisateur) récupéré depuis le SI ne contient pas cn=from.si.ressources.carte,ou=groups,dc=univ-ville,dc=fr

memberOf étant lui-même récupéré via ldap en l'ajoutant dans les attributs ldap à récupérer

```
<bean id="ldapUserInfoService" class="org.esupportail.sgc.services.userinfos.LdapUserInfoService" p:order="2">
  <property name="eppnFilter" value=".*@univ-ville\.fr"/>
  <property name="ldapTemplate" ref="ldapTemplate"/>
  <property name="sgcParam2ldapAttr">
    <map>
      ....
      <entry key="memberOf" value="memberOf"/>
    </map>
  </property>
</bean>
```

cardIdsService :

Permet de configurer la génération d'identifiants qui pourront être codés dans la carte par esup-nfc-tag :

- pour du contrôle d'accès par exemple
- ou pour générer l'identifiant de carte crous si on n'opte pas pour l'usage de carte pré-encodé crous et qu'on souhaite qu'esup-sgc et esup-nfc-tag se chargent de cet encodage. Aussi dans ce cadre crousEncodeEnabled à true permet de spécifier que l'application CROUS doit être écrite lors de l'encodage des cartes.

```
        <bean id="cardIdsService" class="org.esupportail.sgc.services.cardid.CardIdsService">
                <property name="cardIdServices">
                        <list>
                                <bean class="org.esupportail.sgc.services.cardid.
ComueNuAccessControlCardIdService">
                                        <property name="appName" value="access-control"/>
                                        <property name="idCounterBegin" value="XXXXXXXXXXXXXXX"/>
                                        <property name="postgresqlSequence" value="card_sequence"/>
                                </bean>
                                <bean class="org.esupportail.sgc.services.cardid.ComueNuBuCardIdService">
                                        <property name="appName" value="bu"/>
                                </bean>
                                <bean class="org.esupportail.sgc.services.cardid.CnousCardIdService">
                                        <property name="appName" value="crous"/>
                                        <property name="idCounterBegin" value="XXXXXXXX"/>
                                        <property name="postgresqlSequence" value="crous_smart_card_sequence"/>
                                        <property name="crousEncodeEnabled" value="false"/>
                                </bean>
                        </list>
                </property>
        </bean>
```

La propriété idCounterBegin permet de démarrer vos numéros à partir d'une valeur différente de 0.

Mettre 1010000000000000 (par exemple) à ce idCounterBegin pour vos numéros de contrôle d'accès présente plusieurs intérêts :

- vous n'avez pas à vous poser d'éventuelles questions sur le padding électronique réalisé sur vos cartes : tous vos numéros auront la même longueur (sans padding).
- vos numéros supporteront de fait l'attribution de plages de numéros pour un fonctionnement multi-établissements avec multi-instances : il vous suffira en effet de proposer aux autres établissements de configurer leur idCounterBegin par 1020000000000000 puis par 1030000000000000 etc.

C'est via ce même procédé que esup-sgc supporte et respecte les plages d'identifiants crous/izly lors de l'encodage crous/izly ; on met par exemple idCounterBegin 12300000 si le crous/izly nous a fourni une plage (avec clef sam associée) qui démarre à 12300000
Rappel : le crous/izly encourage fortement les établissements à acheter des cartes pré-encodées crous/izly auprès du détenteur du marché national des cartes multiservices pour l'ESR (cf la page du CNCEU - Comité National de la Carte d'Etudiant et de ses Usages - à cet effet), et c'est ce que nous conseillons de faire également.

EsupNfcTagService :

Pour spécifier l'adresse du serveur esup-nfc-tag. Le SGC déclare et contrôle ses périphériques d'encodage (esup-scg-client) avec esup-nfc-tag (applicationName correspond à l'application créée dans esup-nfc-tag)

```
<bean id="esupNfcTagService" class="org.esupportail.sgc.services.EsupNfcTagService">
 <property name="restTemplate" ref="restTemplate"/>
 <property name="webUrl" value="http://esup-nfc-tag.univ-ville.fr"/>
 <property name="applicationName" value="Ecriture SGC"/>
 <property name="location" value="Encodage ESUP SGC"/>
</bean>
```

LdapValidateService

Le SGC peut transmettre des données au LDAP lorsque la carte est activée. Dans le bean ldapValidateService il est possible de paramétrer deux types de clés : Simple ou multivaluée (ldapCardIdsMappingValue, ldapCardIdsMappingMultiValues)

Les valeurs transmissibles sont : %csn%, %reverse_csn% (le csn retourné par paires), %access-control% (ex : identifiant contrôle d'accès), %photo%.

```
<bean id="ldapValidateService" class="org.esupportail.sgc.services.ldap.LdapValidateService">
 <property name="ldapTemplate" ref="ldapTestTemplate"/>
 <property name="peopleSearchFilter" value="(eduPersonPrincipalName={0})"/>
 <property name="ldapCardIdsMappingMultiValues">
  <map>
   <!-- Exemple clé multi-valuée -->
   <entry key="supannRefId">
    <list>
     <value>{ISO15693}%csn%</value>
     <value>{LEOCARTE:ACCESS-CONTROL}%access-control%</value>
    </list>
   </entry>
   <!-- Exemple clé multi-valuée -->
   <entry key="autreExemple">
    <list>
     <value>%reverse_csn%@ISO15693</value>
    </list>
   </entry>
  </map>
 </property>
 <property name="ldapCardIdsMappingValue">
  <map>
    <!-- Exemple clé simple -->
    <entry key="jpegPhoto" value="%photo%"/>
  </map>
 </property>
</bean>
```

PapercutValidateService

Le SGC peut pousser le CSN de la carte sur le gestionnaire d'impression papercut (https://www.papercut.com/).

Voici un exemple de configuration pour activer une telle fonctionnalité

```
<bean id="papercutValidateService" class="org.esupportail.sgc.services.papercut.PapercutService">
  <property name="authToken" value="mon-auth-token-papercut-positionne-dans-conf-avancee" />
  <property name="server" value="papercut.univ-ville.fr" />
  <property name="scheme" value="https" />
  <property name="port" value="443" />
  <property name="accountName" value="" />
  <property name="papercutUidFromEppnRegex" value="([^@]*)@.*"/>
  <property name="cardNumberAttribute" value="card-number"/>
</bean>
```

La propriété papercutUidFromEppnRegex permet de construire l'uid (identifiant) utilisé dans papercut depuis l'eppn : ([^@]*)@.* permet ainsi de sélectionner le groupe (1er groupe) correspondant à toto dans toto@univ-ville.fr par exemple.

La propriété accountName est à ignorer ; elle ne sert actuellement pas.

Par défaut papercutUidFromEppnRegex correspond à (.*) et correspond donc à utiliser comme identifiant papercut l'eppn directement.

L'implémentation de org.esupportail.sgc.services.papercut.PapercutService met à jour la propriété card-number de l'utilisateur si celui-ci existe dans papercut, sinon il ne fait rien (il ne le crée pas).

RestValidateService

Le SGC propose une implémentation ValidateService appelant un service REST à chaque activation ou désactivation de carte.

Couplé à l'API esup-sgc permettant de récupérer des informations utilisateurs et de cartes du SGC (cf FAQ), celà peut vous permettre d'intégrer élégamment et de manière synchrone une brique de votre système d'information encore non intégrée par défaut par esup-sgc, cela sans nécessité de coder dans esup-sgc lui-même.

On peut ici imaginer un service rest répondant derrière http://localhost/cgi-bin/validate-example-sgc.py - on insérera la configuration suivante dans applicationContext-services.xml pour activer celà (notez que le applicationContext-services.xml par défaut ne propose pas un tel exemple).

```
<bean id="myRestValidateService" class="org.esupportail.sgc.services.RestValidateService">
        <property name="restTemplate" ref="restTemplate"/>
        <property name="validateRestUrl" value="http://localhost/cgi-bin/validate-example-sgc.py?eppn=%s&amp;
csn=%s"/>
        <property name="invalidateRestUrl" value="http://localhost/cgi-bin/validate-example-sgc.py?eppn=%s&amp;
csn=%s"/>
</bean>
```

Exemple d'un tel script cgi python validate-example-sgc.py est donné ici :

```
#!/usr/bin
/python

# -*- coding: utf-8 -
*-

import cgi
import requests

arguments = cgi.FieldStorage()
eppn = arguments['eppn'].value

print "Content-Type: text/plain"
print ""
print eppn

resp = requests.get(url = 'https://esup-sgc.univ-ville.fr/wsrest/api/get?eppn=%s' % eppn)

csn = ''
acccesControlId = ''
data = resp.json()
for user in data :
    for card in user['cards'] :
        if card['etat'] == 'ENABLED' :
            csn = card['csn']
                        acccesControlId = card['desfireIds']['access-control']
open_file = open('/opt/export-sgc/a.txt', 'a+b')
open_file.write('card enabled for %s : csn -> %s, access-control : %s\n' % (eppn, csn, acccesControlId))
open_file.close()
print 'OK'
```

applicationContext-crous.xml

ApiCrousService :

Permet d'activer l'envoi de données au CROUS via l'api

```
<bean id="apiCrousService" class="org.esupportail.sgc.services.crous.ApiCrousService">
 <property name="enable" value="false"/>
 <property name="webUrl" value="https://api-pp.nuonet.fr" />
 <property name="appId" value="xxxxxxxxxxxx" />
 <property name="appSecret" value="yyyyyyyyyyyy" />
 <property name="restTemplate" ref="restTemplate" />
</bean>
```

Pour récupérer appId et appSecret la procédure est la suivante :

1. se rendre sur https://developers.lescrous.fr
2. s'authentifier avec le compte utilisateur ; si votre login n'est pas un email, ne pas tenir compte du placeholder "email" dans le 1er champ, il faut mettre le "login" (qui était utilisé par votre application si vous aviez configuré votre esup-sgc avant décembre 2020, demandez un compte au crous:cnous sinon).
3. cliquer sur "Applications" dans le menu latéral puis "Ajouter" dans le menu horizontal
4. remplir le formulaire (nom, description)
5. BIEN NOTER LES identifiants appId et appSecret car ils ne sont pas conservés en clair après validation : les rentrer dans le fichier applicationContext-crous.xml

EsistCrousService :

Fait référence au fichier XML de calcul des tarifs CROUS

ApiEscrService :

Permet de configurer l'envoi des données à l'api ESC (Eropean Student Card) voir : http://europeanstudentcard.eu/

```
<bean id="europeanStudentCardService" class="org.esupportail.sgc.services.esc.ApiEscrService">
 <property name="enable" value="false"/>
 <property name="webUrl" value="http://api-sandbox.europeanstudentcard.eu/v1" />
 <property name="key" value="xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx" />
 <property name="restTemplate" ref="restTemplate" />
 <property name="countryCode" value="FR"/>
 <property name="picInstitutionCode" value="000000000"/>
 <!--
  Type of cards. Possibles values are :
   1 - passive card, with no electronic
   2 - Smartcard without European common data zone
   3 - Smartcard with European common data zone
   4 - Smartcard on which application may be installed by service providers
  -->
 <property name="cardType" value="2"/>
</bean>
```

applicationContext-paybox.xml

Ce fichier permet le paramétrage du module paybox dans le cas d'un renouvellement de carte payant.

Dans les faits, les configurations sont en fait reportées et donc à saisir dans le fichier paybox.properties situé dans le même répertoire :

- paybox.prod.site, paybox.prod.rang, paybox.prod.identifiant, paybox.prod.devise et paybox.prod.hmacKey doivent correspondre à vos paramètres de votre paybox (HMAC). Les paramètres donnés par défaut correspondent au paramètre du compte paybox générique de test.
- paybox.prod.responseUrl doit correspondre à l'url publique de votre sgc (virutalhost) : https://esup-sgc-demo.univ-rouen.fr par exemple pour le site de démonstration d'esup-sgc hébergé à l'Université de Rouen Normandie.

Au niveau de applicationContext-paybox.xml lui-même, pour la production, vous devrez supprimer la première url de la liste des payboxActionUrls, celle-ci étant l'url de preprod (test) de paybox.

Au niveau de applicationContext-paybox.xml on peut également donner un format de numéro de commande paybox, celui-ci doit être garanti unique au niveau de votre compte paybox. Par défaut, c'est 'LEOCARTE-NEW@@%eppn%@@%montantAsCents%-%date%' Vous pouvez le surcharger en mettant par exemple :

```
<property name="numCommandeFormat" value="LEOcarte_new-%supannEmpId%%supannEtuId%-%email%-%montantAsCents%-%
date%"/>
```

applicationContext-access-control.xml

Permet de paramétrer le chemin d'export des informations pour le controle d'accès. Formats disponibles:

- P2S
- TIL
- SYNCHRONIC

applicationContext-security.xml

Comme pour la configuration esup-nfc-tag, il s'agit de mapper les groups sur les rôles proposés par le SGC :

```
<util:map id="sgcMappingGroupesRoles">
        <beans:entry key="cn=for.esup-sgc.admin,ou=groups,dc=univ-ville,dc=fr" value="ROLE_ADMIN" />
        <beans:entry key="cn=for.esup-sgc.super-manager,ou=groups,dc=univ-ville,dc=fr" value="
ROLE_SUPER_MANAGER" />
        <beans:entry key="cn=for.esup-sgc.livreur,ou=groups,dc=univ-ville,dc=fr" value="ROLE_LIVREUR" />
        <beans:entry key="cn=for.esup-sgc.updater,ou=groups,dc=univ-ville,dc=fr" value="ROLE_UPDATER" />
        <beans:entry key="cn=for.esup-sgc.consult,ou=groups,dc=univ-ville,dc=fr" value="ROLE_CONSULT" />
        <beans:entry key="cn=for.esup-sgc.user,ou=groups,dc=univ-ville,dc=fr" value="ROLE_USER" />
        <beans:entry key="cn=from.apogee.dossier-nok,ou=groups,dc=univ-ville,dc=fr" value="
ROLE_USER_NO_EDITABLE" />
        <beans:entry key="cn=from.esup-sgc.users-cards-3years-student,ou=groups,dc=univ-ville,dc=fr" value="
ROLE_USER_RENEWAL_PAYED" />
</util:map>
```

applicationTasksContext.xml

Les groupes/rôles et champs utilisateurs sont récupérés (synchronisés) depuis le Système d'Information via les configurations précédentes et stockés dans la base de données PostgreSQL d'esup-sgc, celà :

- à chaque authentification de l'usager lui-même
- 'régulièrement' via un CRON interne à ESUP-SGC

Ce fichier applicationTasksContext.xml vous permet de régler la synchronisation régulière (notez que seule la synchronisation des groupes est en fait active par défaut !), une configuration de production pourra être celle-ci :

```
<task:scheduled-tasks scheduler="sgcScheduler">
  <task:scheduled ref="ldapGroup2UserRoleService" method="syncAllGroupsOnDb" fixed-delay="300000" initial-
delay="10000"/>
  <task:scheduled ref="resynchronisationService" method="synchronizeAllUsersInfos" fixed-delay="21600000"
initial-delay="10000"/>
</task:scheduled-tasks>
```

Les 'delay' sont donnés en ms : aussi ici on lance les premières synchronisation 10 secondes après le démarrage de l'application, puis on resynchronize les groupes toutes les 5 minutes et les champs de tous les utilisateurs toutes les 6 heures.

> **Synchronisation**
>
> C'est aussi via cette synchronisation que les cartes vont devenir caduques, que les tarifs vont être mis à jour au besoin au niveau du CROUS, que les changements dans les dates de fin vont être prises comptes, etc.

Voyez la question 'Comment sont synchronisés les données utilisateur ?' sur la FAQ pour plus d'informations.

Elements configurables dynamiquement via IHM

Certains paramètres sont modifiables directement depuis l'interface d' ESUP-SGC, cela pour éviter un redémarrage du service. Il faut se rendre dans l'onglet "Admin -> Configuration".

Voici un extrait des paramêtres que l'on peut trouver

Key	Description	Type
MODE_LIVRAISON	Permet d'afficher ou non tout ce qui concerne la livraison de carte.	BOOLEAN

MODE_BORNES	Affiche ou non ce qui concerne les bornes de mise à jour.	BOOLEAN
QRCODE_ESC_ENABLED	Si "false", le QRCODE contient l'eppn. Si "true", il contient l'identifiant dans le cadre de la carte européenne	BOOLEAN
ENABLE_AUTO	Si true, la carte se met directement à l"état "ENABLED" (activé) à la fin de l"encodage, sinon la carte se met à l"état "ENCODED" (encodé).	BOOLEAN
SYNCHRONIC_EXPORT_CSV_FILE_NAME	Nom du fichier CSV Synchronic	TEXT
TIL_EXPORT_CSV_FILE_NAME	Nom du fichier CSV d'export Til	TEXT
P2S_EXPORT_CSV_FILE_NAME	Nom du fichier d'export P2S	TEXT
QRCODE_FORMAT	Format de l'image du QRCODE utilisé pour l'impression. (SVG ou PNG)	TEXT
DISPLAY_FORM_EUROPEAN	Dans le formulaire de demande de carte, population pour laquelle on affiche la partie concernant l'adhésion au projet de carte européenne. E=Etudiant, I=Invité, P=Personnel (ex: IP)	TEXT
MAIL_NO_REPLY	Adresse 'From' des mails envoyés à partir de l'application.	TEXT
MAIL_LISTE_PRINCIPALE	Adresse mail à laquelle sont adressés en copie les mails automatiques de demande de cartes ou lors de l'activation/réactivation de carte.	TEXT

ESUP-NFC-TAG-SERVER

L'installation d'ESUP-NFC-TAG-SERVER est documentée sur cette page ESUP-NFC-TAG-SERVER.

Cette partie traite de la configuration d'Esup-Nfc-Tag-Server en adéquation avec l'installation d'Esup-SGC.

La configuration se fait par les fichiers suivants :

- src/main/resources/META-INF/spring/applicationContext-custom.xml
- src/main/resources/META-INF/spring/applicationContext-desfire.xml (voir Configuration Desfire avancée)
- src/main/resources/META-INF/spring/applicationContext-security.xml

applicationContext-custom.xml

Le fichier applicationContext-custom.xml permet de configurer les différentes applications avec lesquelles esup-nfc-tag va communiquer.

Dans le cadre du SGC il faudra à minima configurer la connexion avec esup-sgc à l'aide de la configuration suivante :

```
<bean id="csnAuthConfig" class="org.esupportail.nfctag.service.api.impl.CsnAuthConfig">
 <property name="description" value="Authentification CSN"/>
</bean>
<bean id="esupSgcWriteExtApi" class="org.esupportail.nfctag.service.api.impl.AppliExtRestWs">
 <property name="isTagableUrl" value="https://esup-sgc-test.univ-ville.fr/wsrest/nfc/isTagable"/>
 <property name="validateTagUrl" value="https://esup-sgc-test.univ-ville.fr/wsrest/nfc/validateTag"/>
 <property name="getLocationsUrl" value="https://esup-sgc-test.univ-ville.fr/wsrest/nfc/locations"/>
 <property name="description" value="Web Service Write Esup SGC"/>
</bean>
<bean id="tagIdCheckApiEsupSgc" class="org.esupportail.nfctag.service.api.impl.TagIdCheckRestWs">
 <property name="tagIdCheckUrl" value="https://esup-sgc-test.univ-ville.fr/wsrest/nfc/tagIdCheck"/>
 <property name="idFromEppnInitUrl" value="https://esup-sgc-test.univ-ville.fr/wsrest/nfc/idFromEppnInit"/>
 <property name="description" value="via Esup SGC"/>
</bean>
```

en remplacant les liens par ceux pointant vers votre instance d'esup-sgc

applicationContext-desfire.xml

Dans le ficher applicationContext-desfire.xml on trouve la structure et les données qui seront écrites sur la carte lors de l'encodage. Par défaut la configuration de Desfire est vide, aucune application ne sera écrite sur la carte.

```
<bean id="desfireAuthConfigComueWriteEsupSgc" class="org.esupportail.nfctag.service.api.impl.DesfireWriteConfig"
>
                <property name="desfireTag" ref="tagName" />
                <property name="description" value="Ecriture ESUP SGC"/>
</bean>


<bean id="tagName" class="org.esupportail.nfctag.beans.DesfireTag" p:formatBeforeWrite="false" p:key="
0000000000000000" p:keyType="DES" p:keyVersion="00">
</bean>
```

Voir la documentation avancée pour l'ajout d'application Desfire sur la carte.

applicationContext-security.xml

Mapping des groupes

Dans le fichier applicationContext-security.xml il faut modifier le mapping des groupes pour l'attribution des rôles Admin et Supervisor en precisant le cn des groupes concernés. Attention, la chaine est sensible à la casse :

```
<beans:bean id="authUserDetailsService" class="org.esupportail.nfctag.security.
ShibAuthenticatedUserDetailsService">
 <beans:property name="mappingGroupesRoles">
  <beans:map>
   <beans:entry key="cn=for.esup-nfc-tag.admin,ou=groups,dc=univ-ville,dc=fr" value="ROLE_ADMIN" />
   <beans:entry key="cn=for.esup-nfc-tag.supervisor,ou=groups,dc=univ-ville,dc=fr" value="ROLE_SUPERVISOR" />
  </beans:map>
 </beans:property>
</beans:bean>
```

Sécurité des WS REST /wsrest

Dans ce même fichier se trouve les IP des machines ayant le droit d'accéder au Web Service d'esup-nfc-tag-server :

```
<intercept-url pattern="/wsrest/**" access="hasIpAddress('127.0.0.1')" /> <!-- Esup-SGC IP Address -->
```

La sécurité de ces accès Web Service se fait en effet simplement par ce filtrage IP. Aussi ici, vous pouvez mettre l'IP d'esup-sgc notamment (vous pouvez mettre plusieurs IP en utilisant 'or') :

```
<intercept-url pattern="/wsrest/**" access="hasIpAddress('127.0.0.1') or hasIpAddress('192.168.1.2')" /> <!--
Esup-SGC IP Address -->
```

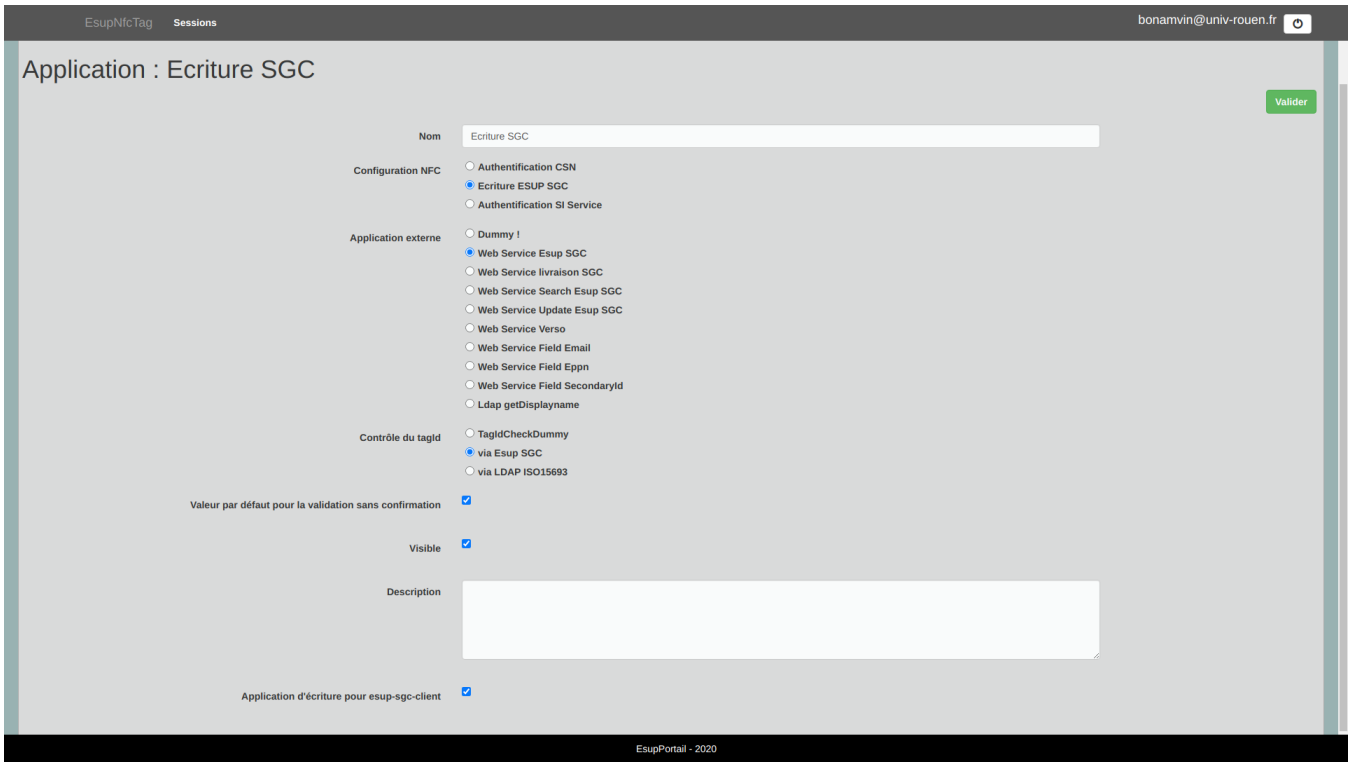
Configuration de l'application dans l'IHM

ESUP-NFC-TAG est multi-service. Il faut donc déclarer les applications, auxquelles il peut s'adresser, au niveau de l'IHM.

Dans la section « Application » « +Ajouter une application »

On retrouve les éléments configurés dans applicationContext-custom.xml. Pour l'encodage des cartes par le SGC, nous allons créer une application nommée "Ecriture SGC" avec les paramètres suivants:

- Nom : « Ecriture SGC »
- Configuration NFC : « Ecriture ESUP SGC » (spécifié dans applicationContext-desfire.xml)
- Application externe : « Web Service Write Esup SGC »
- Contrôle du tagId : « via Esup SGC »
- Valeur par défaut pour la validation sans confirmation : true
- Visible : true
- Application d'écriture pour esup-sgc-client: true



Voir aussi la configuration des applications optionnelles : Applications ESUP-SGC / ESUP-NFC-TAG optionnelles