Intégration d'ESUP-SMSU-API dans Apereo CAS

Disponible dans la **branche master d'esup-smsu-api sur github**, cette possibilité sera effectivement présente dans la prochaine version d'esup-smsu-api (la version qui suit la 3.0.0).

Côté CAS, nous avons testé et validé le bon fonctionnement de l'ensemble avec un **CAS version 6.0.8**

Un serveur Apereo CAS peut utiliser ESUP-SMSU-API pour envoyer des SMS.

ESUP-SMSU-API implémente en effet la méthode REST du service "SMS Messaging" d'Apereo CAS (depuis la version 5.1 de CAS) qui peut permettre à CAS d'envoyer des SMS.

CF https://apereo.github.io/cas/6.0.x/notifications/SMS-Messaging-Configuration.html#rest

Ce mécanisime peut notamment permettre de proposer une Authentifcation Multi Facteurs (MFA) via SMS dans CAS uniquement par configuration (sans ajouter/coder un plugin CAS supplémentaire).

La mise en place de cette possibilité implique de positionner dans le fichier de configuration CAS cas.properties les choses suivantes :

```
# Récupération du numéro de mobile depuis ldap pour le mfa-simple au travers duquel on utilise ESUP-SMSU
...
cas.authn.attributeRepository.ldap[0].attributes.mobile = mobile
...


##############################

## ESUP-SMSU
##

##############################

cas.smsProvider.rest.url=http://smsu-api.univ-ville.fr/apereo-
cas

cas.smsProvider.rest.basicAuthUsername=smsuApiApereoCasAccount
cas.smsProvider.rest.basicAuthPassword=motDePasse


##############################

## MFA
##

##############################

cas.authn.mfa.globalFailureMode=OPEN
cas.authn.mfa.groovyScript=file:/etc/cas/config/mfaGroovyTrigger.groovy

cas.authn.mfa.simple.name=smsu
cas.authn.mfa.simple.order=0
cas.authn.mfa.simple.timeToKillInSeconds=300

cas.authn.mfa.simple.sms.from=Université EsupPortail
cas.authn.mfa.simple.sms.text=Bonjour, voici le code SMS requis pour votre authentification CAS : %s
cas.authn.mfa.simple.sms.attributeName=mobile
```

Ne pas oublier de mettre dans le build.gradle de CAS :

```
compile "org.apereo.cas:cas-server-support-simple-mfa:${project.'cas.version'}"
```

Confère la ligne 'cas.authn.mfa.groovyScript=file:/etc/cas/config/mfaGroovyTrigger.groovy' on décide ici d'activer le MFA via un script groovy, c'est ce qui parait le plus souple à réaliser ici.

Les modifications sur le script sont pris en compte immédiatement.

Le paramètre 'cas.authn.mfa.globalFailureMode=OPEN' indique que si un problème survient autour du MFA, l'authentification est validée simplement.

Un exemple d'un tel script est donnée dans la documentation CAS, on en donne ici un autre :

```
import java.util.*

class SampleGroovyEventResolver {
    def String run(final Object... args) {
        def service = args[0]
        def registeredService = args[1]
        def authentication = args[2]
        def httpRequest = args[3]
        def logger = args[4]

        def mobile = authentication.principal.attributes.mobile

        logger.info("ip : [{}]", httpRequest.getRemoteAddr())
        logger.info("mobile : [{}]", mobile)
        logger.warn("registeredService.id : [{}]", registeredService.id)

        // 10, 11 et 12 des ids de service spécifiques sur lesquels on souhaite faire du MFA pour ceux :
            //  * qui sont à lextérieur de l'université (adresse IP extérieure)
            //  * et ont un numéro de mobile d'informé dans le ldap
        if ((int)registeredService.id in [10, 11, 12] && !httpRequest.getRemoteAddr().startsWith("10.") &&
mobile!=null && mobile.size()>0 && mobile.get(0).startsWith("0")) {
            logger.warn("mfa for [{}] !", authentication.principal.id)
            return "mfa-simple"
        }

        return null
    }
}
```