

Configuration

- [Récupération des sources](#)
- [Fichiers de configuration](#)
 - [application.yml](#)
 - [global](#)
 - [tomcat](#)
 - [spring](#)
 - [session](#)
 - [data, datasource](#)
 - [jpa](#)
 - [ldap](#)
 - [mail](#)
 - [security](#)
 - [ldap](#)
 - [mail](#)
 - [sms](#)
 - [dss](#)
 - [fs \(file system\)](#)
 - [pdf](#)
 - [security](#)
 - [server](#)
 - [sign](#)
 - [logging](#)
 - [springdoc](#)
 - [logback.xml](#)
 - [src/main/resources/i18n/messages.properties](#)
- [Logos et filigrane](#)

Récupération des sources

Lorsque tous les [Prérequis](#) sont installés et [PostgreSQL](#) configuré, vous pouvez récupérer les sources.



A partir de cette étape vous n'êtes plus en **root**. Ici l'utilisateur courant est le compte local **esup** :

```
cd /opt/  
sudo mkdir esup-signature  
sudo chown esup:esup esup-signature/ -R  
git clone https://github.com/EsupPortail/esup-signature.git  
cd esup-signature
```



Le code est cloné dans le dossier `./esup-signature`, le dépôt est positionné sur la branche **master**. Pour toutes les informations relative à l'exploitation et à la mise à jour du code voir le page dédié ici : [Exploitation coté serveur](#)

Fichiers de configuration



Le fichier application.yml présent dans les sources est un exemple de configuration basée sur environnement comportant un LDAP et un gestionnaire de groupes (grouper utilisé par l'Université de Rouen en l'occurrence). Les mentions de "for.esup-signature.xxxx" correspondent aux noms des groupes fournis par grouper. Cela ne veut pas dire que grouper soit obligatoire ni même que les noms de groupes doivent respecter ce format.

Lorsque vous utiliser maven pour compiler ou lancer l'application c'est le fichier situer dans `src/main/resources/application.yml` qui est pris en compte. Il est possible de placer le fichier `application.yml` ailleurs sur le système de fichier en précisant son emplacement à l'aide de l'option `-Dspring.config.location=/<DIR>/application.yml` lors de l'utilisation de la commande `mvn`

application.yml

La configuration principale d' esup-signature se fait au travers application.yml

Pour commenter une ligne il faut ajoute un # devant.



Il faut faire attention à l'indentation lors de la modification du fichier. Une mauvaise indentation peut faire échouer la compilation

De plus, le fichier doit impérativement être encodé en UTF-8 sinon la compilation peut échouer; Exemple d'erreur : [ERROR] Failed to execute goal org.apache.maven.plugins:maven-resources-plugin:3.2.0:resources (default-resources) on project esup-signature: Input length = 1

global

Dans la plupart des cas il s'agit ici de modifier root-url par l'adresse de votre esup-signature. Vous pouvez aussi configurer le système d'archivage et activer le switch user.



Pour des tests en local, il est possible d'utiliser <http://localhost:8080> pour root-url

```
global:
    root-url: https://esup-signature.univ-ville.fr                                # Adresse d'accès à
votre instance
    domain: univ-ville.fr
    nexu-url: https://localhost:9895
    nexu-version: 1.23-SNAPSHOT
    nexu-download-url: /downloads/nexu-bundle.zip
    hide-wizard:
false                                                                                              #
Désactiver le bouton "Assistant de création de demande"
    hide-auto-sign:
false                                                                                                # Désactiver le
bouton "Auto signature"
        hide-send-sign-request: false                                                                #
Désactiver le bouton "Demander une signature"
        hide-wizard-except-
roles:                                                                                               # Rôles faisant exception
à la règle hide-wizard précédente
        hide-auto-sign-except-roles:                                                                  #
Rôles faisant exception à la règle hide-auto-sign précédente
        hide-send-sign-except-roles:                                                                    #
Rôles faisant exception à la règle hide-send-sign précédente
#     archive-uri: smb://serveur_fichier/archives                                                    # Chemin pour
l'archivage des documents
#     delay-before-cleaning : 0                                                                      #
Délai en jours après signature pour archivage et nettoyage des documents (désactivé si commenté)
    enable-su:                                                                                        # Activer
ou non le switch user
        enable-splash:
true                                                                                                  # Activer
ou non le message d'accueil de la première connexion
            application-email: esup.signature@univ-ville.fr                                          # Adresse email
du support qui apparaîtra dans l'aide
                hours-before-refresh-notif: 24                                                         #
Nombre d'heures entre deux relances utilisateur
                share-mode:
```

```

Valeur de 0 à 3 : 0 = délégations désactivées, 1 = force le mode signature du délégué, 2 = force signature du
mandant, 3 = choix du mode possible par l'utilisateur
    return-to-home-after-sign: false #
Forcer le retour à la page d'accueil après signature
    infinite-scrolling: true #
Activer l'infinite scrolling sur le tableau de bord, sinon bascule sur de la pagination
    signed-suffix:
"_signé" # suffix ajouté
au fichiers signés
    naming-template: "[title]" #
Template de renommage des fichiers
    enable-scheduled-cleanup: false
# Activer ou non l'archivage et le nettoyage automatique. false par défaut
    trash-keep-delay:
-1 # Délai de
conservation dans la corbeille en jours (-1 non actif)
    disable-cert-storage:
false # Activer/Désactiver la
possibilité de stocker des certificats utilisateurs
    nb-days-before-deleting:
-1 # Nombre de jours après
alerte pour suppression des demandes en attente (-1 non actif)
    nb-days-before-warning:
-1 # Nombre de jours avant
alerte de suppression pour les demandes en attente (-1 non actif)
    enable-captcha:
false # Activer
/Désactiver la detection de robot à la connexion ;- )
    max-upload-size:
52428800 # Taille maximum des
uploads de fichiers en bytes
    only-pdf:
false #
True : restreindre l'upload aux seuls PDF
    export-attachements: true #
Exporter les pièces jointes (si actif, l'export sera un dossier contenant le document signé ainsi que les PJ)
    seal-certificat-driver: /lib/pkcs11/libIDPrimePKCS11.so # Pilote du certificat
cachet dans le cas d'un PKCS11
    seal-certificat-file: /opt/cert.p12 #
Emplacement du certificat cachet dans le cas d'un PKCS12
    seal-certificat-pin: *****
# Code pin du certificat cachet
    seal-certificat-type: PKCS11 #
Type du certificat PKCS11 ou PKCS12
    seal-all-docs:
false # Appliquer le
cachet sur toutes les demandes terminées
    shib-users-domain-white-list: #
Whitelist des domaines autorisés à obtenir le ROLE_USER pour les connexions Shibboleth
    - univ-ville.fr
    - inv-univ-ville.fr
    send-postit-by-email:
false # Envoyer un email au
créateur de la demande lors de l'ajout d'un postit
    send-creation-mail-to-viewers: false # Envoyer un
email aux observateurs à la création d'une demande
    sms-required:
true # Imposer
la double authentification par SMS pour les externes
    csv-quote:
"\\" #
Quote CSV
    csv-separator:
";" # Séparateur
CSV
    otp-validity:
1 #
Durée de validité des liens de OTP en minutes
    authorized-sign-types: certsign, nexuSign # Liste des types
de signature autorisés
    watermark-for-externals: true #

```

Activer/Desactiver le watermark pour les utilisateurs externes

Pour la gestion des rôles voir : [Documentation administrateur#Gestiondesr%C3%B4les](#)



Détails concernant le système de renommage des fichiers :

Le modèle est construit à l'aide d'attributs entre crochets.

default : [title]

Les attributs disponibles sont :

- [title] : titre du document original
- [id] : identifiant du parapheur
- [workflowName] : nom du circuit
- [user.name] : nom prénom de l'utilisateur courant
- [user.eppn] : eppn de l'utilisateur courant
- [user.initials] : initiales de l'utilisateur courant
- [UUID] : un identifiant unique
- [order] : le numéro d'ordre de création pour un même circuit
- [timestamp] : timestamp sous forme de long
- [date-fr] : date dd/MM/yyyy hh:mm
- [date-en] : date yyyy-MM-dd hh:mm

tomcat

Ce paramètre permet de spécifier le port ajp dans le cas où l'application est démarrée directement (en lançant directement esup-signature.war ou en utilisant mvn springboot:run par exemple)

Par défaut ces lignes sont commentées, cela doit être ainsi lorsque l'application est démarrée par un serveur Tomcat.

```
tomcat:
  ajp:
    port: 6009
```

spring

session

À ne pas modifier. Permet l'activation de spring-session.

```
session:
  jdbc:
    initialize-schema: always
    save-mode: always
```

data, datasource

Configuration de la base de donnée :

```
datasource:
  driver-class-name: org.postgresql.Driver
  url: jdbc:postgresql://localhost:5432/esupsignature
  password: esup
  username: esupsignature
  jdbc-url: ${spring.datasource.url}
  hikari:
    auto-commit: false
```

jpa

Il faut prêter une attention particulière au paramètre `ddl-auto`. Mode **update** permet la création et la mise à jour de la base de donnée lors du démarrage de l'application ou des tests. Ce mode peut éventuellement être utilisé lors des mises à jour de l'application. Le reste du temps (en production par exemple) ce paramètre peut être positionné sur **validate** qui ne fera qu'exécuter un contrôle de la base de données.



Attention au mode **create** qui, lui, détruit et re-crée la base complète au moment du démarrage de l'application

```
jpa:
  hibernate:
    ddl-auto: update
  properties:
    hibernate:
      dialect: org.hibernate.dialect.PostgreSQLDialect
      format_sql: true
      jdbc:
        lob:
          non_contextual_creation: true
      show_sql: false
      open-in-view: false
```

ldap

Configuration de l'accès ldap pour spring (obligatoire si l'authentification CAS est activée)

```
ldap:
  base: dc=univ-ville,dc=fr
  password: *****
  urls: ldap://ldap.univ-ville.fr
  username: cn=consult,dc=univ-ville,dc=fr
```

mail

Configuration du serveur de mail pour l'envoi des alertes

```
mail:
  host: smtp.univ-ville.fr
```

servlet, thymeleaf, web, mvc

Ne pas modifier, concerne la configuration web de spring

```
servlet:
  multipart:
    enabled: true
    max-file-size: 1280KB
    max-request-size: 1280KB
    resolve-lazily: true
thymeleaf:
  cache: false
  encoding: UTF-8
  mode: HTML
  servlet:
    produce-partial-output-while-processing: false
web:
  resources:
    cache:
      cachecontrol:
        max-age: 1d
        cache-public: true
      static-locations: classpath:/static
mvc:
  static-path-pattern: /**
```

security

Via le protocole oauth2 inclus dans spring-security il est possible de configurer l'authentification avec n'importe quel fournisseur d'identité compatible.

Voici un exemple de configuration via France Connect

```
security:
  oauth2:
    client:
      registration:
        franceconnect:
          provider: franceconnect-idp
          authorization-grant-type: authorization_code
          client-id: <client_id>
          client-secret: <client_secret>
          client-authentication-method: client_secret_post
          redirect-uri: "{baseUrl}/login/oauth2/code/{registrationId}"
          scope:
            - openid
            - family_name
            - given_name
            - email
      provider:
        franceconnect-idp:
          authorization-uri: https://fcp.integ01.dev-franceconnect.fr/api/v1/authorize
          token-uri: https://fcp.integ01.dev-franceconnect.fr/api/v1/token
          user-info-uri: https://fcp.integ01.dev-franceconnect.fr/api/v1/userinfo
          user-name-attribute: sub
          user-info-authentication-method: header
```

Ldap

La configuration Ldap hors spring est spécifique à votre établissement, elle précise les modalités de recherche de vos utilisateur dans l'annuaire

```

ldap:
  search-base:
ou=people
# Base de recherche des utilisateurs, ex : ou=people
  user-object-classes:
inetOrgPerson
# Object classes correspondant aux utilisateurs (un "ou" est appliqué aux valeurs de cette liste)
  group-object-classes:
groupOfNames
# Object classes correspondant aux groupes (un "ou" est appliqué aux valeurs de cette liste)
  ou-object-classes:
organizationalUnit
# Object classes correspondant aux OU (un "ou" est appliqué aux valeurs de cette liste)
  alias-object-classes:
nisMailAlias

  users-search-filter: (&(|(displayName={0}*)(cn={0}*)(uid={0})(mail={0}*)))(mail=*) # Filtre de recherche
des utilisateurs
  group-search-base:
ou=groups
# Base de recherche des groupes, ex : ou=groups
  group-search-filter: member=
{0}
# Filtre utilisé pour rechercher les groupes d'un utilisateur, ex : member={0}
  all-groups-search-filter: cn=*
{0} #
Filtre utilisé pour rechercher des groupes, ex : cn={0}
  all-aliases-search-filter: (mail=
{0}) # Filtre
utilisé pour rechercher des aliases
  user-id-search-filter: (uid=
{0}) #
Le champ dans lequel on trouve le login des utilisateurs, ex : (uid={0})
  user-eppn-search-filter: (eduPersonPrincipalName=
{0}) # Le champ dans lequel on trouve l'eppn des
utilisateurs c'est ce champ qui sera utilisé comme identifiant unique en base, ex : (eduPersonPrincipalName={0})
  user-mail-search-filter: (mail=
{0}) #
Le champ dans lequel on trouve l'email des utilisateurs , ex : (mail={0})
  ou-search-filter: (supannCodeEntite=
{0}) # Requete pour
trouver les OU des utilisateurs (utile seulement pour le pré-remplissage de l'affectation dans les formulaires)
  member-search-filter: (&(uid={0})
({1})) # Filtre
pour contrôler l'appartenance d'un utilisateur à un groupe, ex : &(uid={0})({1})
  members-of-group-search-filter: memberOf=cn={0},ou=groups,dc=univ-ville,dc=fr # Filtre
utilisé pour retrouver les membres d'un groupe, ex : memberOf=cn={0},ou=groups,dc=univ-ville,dc=fr
  eppn-left-part-search-filter: (uid=
{0}) # Permet de
gérer le cas où l'eppn n'est pas construit avec <uid>@<domain>. Il faut donc spécifier le champ dans lequel on
trouve la partie gauche de votre eppn
  mapping-filters-
groups:
# Liste d'attribution de groupe en fonction d'un filtre LDAP
  student : "(eduPersonAffiliation:=student)"
  staff : "(eduPersonAffiliation:=staff)"

```



On peut utiliser **mapping-filters-groups** pour attribuer des groupes à un utilisateur, par exemple : avec mes-users : "eduPersonAffiliation:=member", toutes les personnes issue de ce filtre se verront affectées au groupe "mes-users"

Si dans **mapping-groups-roles** on a mes-users: ROLE_USER, l'utilisateur obtiendra le rôle ROLE_USER.



Attention vos requetes LDAP doivent impérativement être mises entre **parentheses**.

De plus les objectClasses des éléments recherchés sont écrits en dur dans les classes PersonLdap, PersonLightLdap, OrganizationalUnitLdap et AliasLdap. Pour élargir le champ de recherche il faut modifier ces classes directement. Elle sont situées dans le dossier :

src/main/java/org/esupportail/esupsignature/service/ldap/entry

Pour la gestion des rôles voir : [Documentation administrateur#Gestion des rôles](#)

mail

Dans la partie mail (hors spring) vous pouvez paramétrer l'adresse from pour l'envoi de mails

```
mail:
  from: no-reply.esup-signature@univ-ville.fr
```

sms

Cette partie permet de configurer un service d'envoi de sms pour l'utilisation de la fonction One Time Password à destination des personnes extérieures à l'établissement. Des implémentations pour SMSU et pour OVH sont disponibles. Il est possible d'en ajouter en implémentant la classe <https://github.com/EsupPortail/esup-signature/blob/master/src/main/java/org/esupportail/esupsignature/service/interfaces/sms/SmsService.java>

```
sms:
  enable-sms : false
#   service-name: SMSU
#   url: https://smsu-api.univ-ville.fr/
#   username: sms-account
#   password: *****
```

dss

tsp-servers : adresses url des serveurs de temps utilisé pour les horodatages des signatures électroniques



Pour que la signature soit valable, le fournisseur du timestamp doit apparaître dans la liste European Trusted List (EUTL). Sur cette page on trouve une liste de serveurs utilisables : <https://gist.github.com/Manouchehri/fd754e402d98430243455713efada710>. Il faut faire attention à ne prendre que des fournisseurs Adobe ou EUTL.

Par défaut, DSS Signature propose d'utiliser <http://tsa.belgium.be/connect> cependant il semble que le serveur impose un nombre maximum de requetes.

Depuis la version 1.28.9 il est possible de saisir plusieurs urls. Dans ce cas le système essaye chaque adresse jusqu'à obtenir un timestamp.

ks-filename : par défaut le chemin est relatif. Comme dans la configuration logback, si vous voulez spécifier un chemin absolu, le dossier doit avoir été créé avant le premier lancement.

trusted-certificat-url-list : ici vous pouvez ajouter des liens vers les url des certificats non présents dans le journal officiel mais valide dans votre établissement :



Les autres paramètres n'ont pas besoin d'être modifiés

```
dss:
  cache-data-source-driver-class-name: org.hsqldb.jdbc.JDBCDriver
  cache-data-source-url: jdbc:hsqldb:mem:cachedb
  cache-password:
  cache-username: sa
  default-validation-policy: policy/sign-constraint.xml
  server-signing-keystore-filename: validate_service.p12
  server-signing-keystore-password: password
  server-signing-keystore-type: PKCS12
  tsp-servers:
    - http://timestamp.sectigo.com/qualified
    - http://tsa.belgium.be/connect
  ks-filename: oj_keystore.p12
  ks-password: dss-password
  ks-type: PKCS12
  lotl-country-code: EU
  lotl-url: https://ec.europa.eu/tools/lotl/eu-lotl.xml
  oj-url: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=uriserv:OJ.C_.2019.276.01.0001.01.ENG
  country: FR
  state-or-province: Région
  postal-code: XXXXX
  locality: Ville
  check-revocation-for-untrusted-chains:
true
#Débloque la possibilité de signer avec des certificats non eIDas (ex : Sectigo Rénater)
```

Si votre serveur se trouve derrière un forward proxy, vous pouvez ajouter la configuration suivante directement à la racine du fichier de configuration. Cela permet à DSS d'aller chercher les certificats de confiance sur les serveurs européens

```
proxy:
  http-enabled: true
  http-host: localhost
  http-port: 3128
  http-user:
  http-password:
  http-excluded-hosts:
  https-enabled: true
  https-host: localhost
  https-port: 3128
  https-user:
  https-password:
  https-excluded-hosts:
```

fs (file system)

La partie file system (fs) permet de configurer une source principale de donnée pour chaque type de stockage (smb, cmis, vfs). esup-signature ne prend en charge qu'une seule source par type.

```

fs:
  smb-login: esup-
signature                                     # login du
compte autorisé à accéder au partage SMB
  smb-password: *****
  smb-uri-test: smb://smb.univ-ville.fr          #
chemin vers un partage de test. Pour smb il sera possible d'utiliser des chemins absolus
  vfs-test-uri:
/tmp                                           #
chemin vers un repertoire local de test. Pour vfs il sera possible d'utiliser des chemins absolus (FTP compris)
  cmis-uri-test: https://esup-signature.univ-ville.fr/nuxeo # url du chemin vers le
serveur CMIS /\ ce chemin sert pour autant pour les tests que comme référence absolue du serveur
  cmis-login:
Administrator                                # login
autorisé à accéder au serveur CMIS
  cmis-password: Administrator
  cmis-respository-id: default
  cmis-root-path: /default-domain/workspaces    # chemin
vers le dossier accessible par esup-signature

```



Il y a actuellement une confusion concernant la configuration des acces aux stockages externes.

Pour les stockages utilisant de SMB ou VFS, il s'agit de configurer une url de test (pour les test d'intégration). Par la suite, au niveau du paramétrage dans esup-signature, il sera possible de mettre n'importe quel url absolue (pour smb, le compte configuré devra avoir les bons accès)

Pour CMIS, l'attribut cmis-uri-test est mal nommé car il s'agit de l'adresse "fixe" du server CMIS (Nuxeo par ex). Au niveau de l'application il faudra configurer des adresses relatives.

Pour plus d'information voir la page [Gestion des circuits#D%C3%A9finir une source pour les documents](#)

pdf

Paramètres de traitement des PDF

```

pdf:
  convert-to-pdf-a: true
  path-to-g-s: /usr/bin/gs
  pdf-a-level: 2
  pdf-to-image-dpi: 72

```

security

La sécurité est gérée par Spring Security. Il est possible d'activer 3 mécanismes de sécurité : OAuth, Shibboleth et/ou CAS. Pour désactiver une ou l'autre de ces méthodes, il suffit de commenter les lignes qui s'y réfèrent.



Pour une explication détaillée du fonctionnement de la sécurité, voir : [Configuration de la sécurité](#)

Pour CAS :

url : l'adresse de votre serveur CAS

service : l'url de votre esup-signature + /login/cas



CAS nécessite la configuration d'un LDAP. C'est la solution idéale pour une utilisation interne

Pour Shibboleth :

credentials-request-header : attribut dans lequel on trouve les groupes de l'utilisateur

idp-url : adresse de votre IDP Shibboleth

principal-request-header : attribut dans lequel on trouve l'identifiant de l'utilisateur

```
security:
  cas:
    service: https://esup-signature.univ-ville.fr/login/cas
    title: Compte Université (CAS)
    url: https://cas.univ-ville.fr
#  shib:
#    credentials-request-header: MEMBER
#    idp-url: https://idp.univ-ville.fr
#    principal-request-header: REMOTE_USER
#    title: Compte d'un autre établissement (Shibboleth)
#    domains-white-list-url: https://eduspot.renater.fr/eduspot/whitelist-eduspot.txt
  web:
    group-to-role-filter-pattern: <pattern groupe esup-signature>(\w*)           #ici on configure
un pattern permettant de retrouver (discerner) vos groupes dédiés à esup-signature
    mapping-groups-
roles:
    <nom du groupe des administrateurs>: ROLE_ADMIN
    <nom du groupe utilisateur autorisés à accéder à l'application>: ROLE_USER
    ws-access-authorize-ips: 127.0.0.1
#    group-mapping-spel:
#      for.esup-signature.user: "true"
```



group-to-role-filter-pattern: Ce pattern permet de donner automatiquement les rôles correspondants à la partie (w*) du pattern. Prenons le pattern **for.esup-signature.role.(w*)**, si l'utilisateur est dans le groupe for.esup-signature.role.bondecommande, il obtiendra automatiquement le rôle ROLE_BONDECOMMANDE utilisable par la suite pour donner des droits au niveau des formulaires et des circuits.

mapping-groups-roles: Permet de définir des rôles en fonction des groupes de l'utilisateur. **Le ROLE_USER est maintenant obligatoire pour accéder à l'application.** Si constituer un groupe d'utilisateur est difficile, vous pouvez utiliser mapping-filters-groups dans la conf ldap pour constituer des groupes à l'aide de requêtes ldap.

ws-access-authorize-ips : permet de configurer les adresses autorisées à accéder aux web services d' esup-signature. On peut mettre autant d'adresses que souhaité séparées par des virgules ainsi que des plages du genre 192.168.1.0/24. (Exemple : ws-access-authorize-ips : 127.0.0.1, 192.168.1.0/24, 10.54.20.11, etc.)

group-mapping-spel: Sert à outre passer les groupes obtenus via ldap. Ceci est utiles si vous n'avez pas de configuration LDAP (shibboleth seul par exemple). On utilise alors la syntaxe SePL (Spring Expression Language), avec seulement l'attribut **#eppn** pris en compte ainsi que la possibilité de mettre la valeur **"true"**.

Ex: pour ajouter à tout le monde le groupe "for.esup-signature.user" on peut mettre for.esup-signature.user: "true". On peut aussi utiliser la syntaxe suivante pour saisir "en dure" des personnes : for.esup-signature.admin: "#eppn == 'toto@univ-ville.fr' or #eppn == 'titi@univ-ville.fr' "

server

Paramètre du serveur tomcat embarqué dans spring-boot

```

servlet:
  session:
    timeout: 1800 # Durée de session en
secondes
    tracking-modes: cookie
  error:
    include-stacktrace: always
port: 8080
tomcat:
  mbeanregistry:
    enabled: true
  remoteip:
    remote-ip-header: X-Forwarded-For
  basedir: ./tem

```

sign

Vous pouvez ajuster finement les paramètres de signature électronique en particulier le niveau défini par la norme ETSI EN 319 102-1.



Citation documentation DSS

La norme ETSI EN 319 102-1 (cf. [\[R09\]](#)) définit quatre classes de conformité pour répondre au besoin de protéger la validité de la signature dans le temps. Désormais, pour désigner la classe de la signature, on utilisera le mot « niveau ». Suit la liste des profils implémentant les quatre classes définies dans la norme :

- ***AdES_BASELINE_B** : Profil implémentant la classe de signature *Signature de base*
La version la plus basse et la plus simple contenant au moins une valeur de signature, une référence ou une copie du certificat de signature en tant qu'attribut signé, et éventuellement d'autres attributs signés ou non signés.
- ***AdES_BASELINE_T** : Profil implémentant la classe de signature *Signature avec heure*
Un horodatage concernant l'heure de signature est ajouté pour se protéger contre la répudiation.
- ***AdES_BASELINE_LT** : Profil implémentant la classe de signature *Signature avec matériel de validation à long terme*
Tous les éléments ou références aux éléments nécessaires à la validation de la signature sont intégrés pour permettre une vérification future même si leur source d'origine n'est pas disponible. Par exemple, ce niveau doit prouver que le chemin de certification était valide, au moment de la validation de la signature, jusqu'à un point de confiance selon les contraintes de nommage et les contraintes de la politique de certification de la « Signature Validation Policy ».
- ***AdES_BASELINE_LTA** : Profil implémentant la classe de signature *Signature assurant la disponibilité et l'intégrité à long terme du matériel de validation*
En utilisant un horodatage périodique (par exemple chaque année), la disponibilité ou l'intégrité des données de validation est maintenue. La validité pourrait être limitée en raison de l'obsolescence cryptographique des algorithmes, des clés et des paramètres utilisés, ou en raison de l'expiration ou de la révocation du matériel de validation. L' AdES -BASELINE - LTA augmentation ajoute des horodatages supplémentaires pour archiver les signatures de manière à ce qu'elles soient toujours protégées, mais également pour pouvoir prouver que les signatures étaient valides au moment où les algorithmes cryptographiques utilisés étaient considérés comme sûrs. Des données de validation supplémentaires peuvent également être incluses.

Les ajustements sont à opérer ici :

```

sign:
  cades-digest-algorithm: SHA256
  cades-signature-level: CAdES_BASELINE_T
  container-type: ASiC_E
  default-signature-form: XAdES
  padest-digest-algorithm: SHA256
  padest-signature-level: PAdES_BASELINE_T
  password-timeout: 60000
  signature-packaging: ENVELOPED
  xades-digest-algorithm: SHA256
  xades-signature-level: XAdES_BASELINE_T
  sign-with-expired-certificate: false #Débloque la possibilité de signer avec
des certificats expirés (pour les tests)

```

logging

Permet de spécifier l'emplacement pour le fichier de logs ainsi que l'usage (facultatif) d'un fichier de configuration logback (voir chapitre suivant)

```
logging:
  file:
    name: logs/esup-signature.log
  level:
    root: info
    org.esupportail.esupsignature: info
    org.verapdf: error
    org.apache.pdfbox: error
    eu.europa.esig.dss: error
    org.springframework.web.filter.CommonsRequestLoggingFilter: error
#   config: classpath:logback-prod.
xml                                     # permet de
configurer logback via un fichier xml
```

springdoc

Active l'api-doc. Pour permettre l'usage des web service directement depuis swagger, il faut modifier supported-submit-methods avec ["get", "post", "put"]

```
springdoc:
  api-docs:
    enabled: true
  swagger-ui:
    enabled: true
    supported-submit-methods: []
  packages-to-scan: org.esupportail.esupsignature.web.ws
```



Lorsque votre configuration sera terminée, vous devez créer un commit git, ceci afin d'éviter tout problème lors d'une prochaine mise à jour. Les commandes git à lancer:

```
git add .
git commit -m "ma conf de prod"
```

logback.xml

```
<appender name="FILE" class="ch.qos.logback.core.FileAppender">
  <file>logs/esup-signature.log</file>
  <append>true</append>
  <immediateFlush>true</immediateFlush>
  <encoder>
    <pattern>[%-5level] %date{dd/MM/yyyy HH:mm:ss} %logger{35} - %msg%n</pattern>
  </encoder>
</appender>
```

src/main/resources/i18n/messages.properties

Vous pouvez modifier les premières lignes du fichier messages.properties pour spécifier le texte du footer et le titre de l'application

```
application.title = Esup Signature
application.footer=Université de Rouen Normandie
```

Logos et filigrane

Vous avez la possibilité de modifier les logos et le filigrane de la signature(watermark). Pour cela vous devez modifier les fichiers qui se trouvent dans le dossier `src/main/resources/static/images/` :

- Le logo esup-signature correspond aux fichiers `logo.png` et `logo.svg` (le png doit faire maximum 45px de haut si vous ne voulez pas avoir à modifier le css)
- Le logo de l'université visible dans les emails correspond au fichier `logo-univ.png`
- Le filigrane correspond au fichier `watermark.png`. Ses dimensions doivent être de 300*150 ou de 600*300



La suite ici : [Compilation et déploiement](#)