



GUIDE D'INSTALLATION PWM

Installation et configuration



Référence	Installation PWM
Version	1.6
Etat	Confidentiel
Date	07/12/2023
Rédacteur(s)	Cedric Derval

Propriétaire	SYNETIS		
Approbateur		Date	
Vérificateur (s)			
Rédacteur (s)	Cédric DERVAL	Date	17/08/2022

LISTE DE DIFFUSION

Société	Nom	Fonction	E-mail
SwissLife	Jonathan PASQUIER	Chef de Projet	jonathan.pasquier.externe@swisslife.fr

SUIVI DES MODIFICATIONS

Version	Date	Commentaire	Auteur
0.1	17/08/2022	Création du document	Cédric DERVAL
1.0	18/08/2022	Revue du document avec SwissLife et mise à jour en version 1.0	Cédric DERVAL
1.1	23/08/2022	Mise à jour du chapitre SSL	Cédric DERVAL
1.2	24/08/2022	Mise à jour suite installation REC	Cédric DERVAL
1.3	28/07/2023	Ajout des informations de développements custom pour la compatibilité ITDS	Cédric DERVAL
1.4	23/11/2023	Ajout du backup/restore de la configuration	Cédric DERVAL
1.5	01/12/2023	Correction procédure backup/restore	Cédric DERVAL
1.6	07/12/2023	Ajout du chapitre 4.2. Détail des variables du fichier PwmConfiguration.xml	Cédric DERVAL

SOMMAIRE

1. INTRODUCTION.....	4
1.1. OBJECTIFS DU DOCUMENT	4
1.2. PERIMETRE DU DOCUMENT	4
2. PREREQUIS	5
2.1. PREREQUIS RESEAU	5
2.2. SERVEUR PWM	5
2.3. RECAPITULATIF DES PREREQUIS TECHNIQUES	6
2.4. PREREQUIS APPLICATIFS	7
3. CONFIGURATION DU SERVEUR TOMCAT.....	8
3.1. INSTALLATION D'OPENJDK 11	8
3.2. MISE A JOUR DU FICHIER DE CONFIGURATION	8
3.3. IMPORT DU CERTIFICAT SSL (UNIQUEMENT EN DTU).....	9
4. DEPLOIEMENT DE PWM	10
4.1. BUILD DEPUIS LE SHAREPOINT	10
4.2. DETAIL DES VARIABLES DU FICHIER PWMCONFIGURATION.XML	10
4.3. DEPLOIEMENT DE L'APPLICATION	11
4.4. BACKUP/RESTORE DE LA CONFIGURATION	12
4.4.1. En ligne de commande	12
4.4.2. Via l'interface d'administration	12
5. DESCRIPTION DES DEVELOPPEMENTS CUSTOM.....	14
5.1. STRUCTURE DU PROJET	14
5.2. LDAPCHAI	14
5.2.1. AbstractChaiUser.....	15
5.2.2. EdirErrorMap.....	16
5.2.3. EdirEntries	17
5.2.4. OracleDSEErrorMap	18
5.2.5. FailOverWrapper.....	18
5.2.6. JNDIProviderImpl.....	19
5.2.7. checkstyle	21
5.2.8. Pom.xml	22
5.3. PWM-SERVER	22
5.3.1. checkstyle-suppression.xml.....	23
5.3.2. Pom.xml	23
5.3.3. ChangePasswordServlet	23
5.3.4. ChangePasswordServletUtil	27
5.3.5. ForgottenPasswordServlet	28
5.3.6. LDAPAuthenticationRequest	28
5.3.7. PasswordUtility	30
5.4. PWM WEBAPP	35

1. INTRODUCTION

1.1. OBJECTIFS DU DOCUMENT

Le but de ce document est d'expliquer les étapes d'installation et de configuration de la solution PWM sur un serveur d'applications Apache Tomcat.

1.2. PERIMETRE DU DOCUMENT

Ce document contient

- Pour le périmètre Apporteurs B2E/B2B :
 - La procédure de configuration d'Apache Tomcat
 - La procédure de déploiement et de configuration de l'application PWM

2. PREREQUIS

2.1. PREREQUIS RESEAU

- Nom de domaine public pour le service de changement de mot de passe (swisslife.fr)
- Certificat SSL (Secure Sockets Layer) associé au nom de DNS du service de changement de mot de passe (swisslife.fr) :
 - o Les environnements de recette et de production sont en SSL Offload. Les serveurs tomcat sont en HTTP sur le port 8080 et le HTTPS est géré directement sur les VIP.
- Un loadbalancer de répartition sur les serveurs Apache,
 - o avec une adresse IP virtuelle associé au nom de domaine du service d'authentification,
 - o accessible depuis le réseau interne et la zone publique,
 - o Avec une redirection de ports de HTTPS (sur la VIP) vers HTTP/8080 sur les serveurs Tomcat
- Les flux LDAP entre les serveurs PWM et les annuaires ITDS doivent être ouverts
 - o LDAP sur le port 2389 entre PWM et l'ITDS Apporteurs
 - o **LDAPS sur le port 1636 entre PWM et l'ITDS Apporteurs**
- Les flux SMTP entre les serveurs PWM et les serveurs SMTP doivent être ouverts
 - o STARTTLS sur le port 587 entre PWM et les serveurs SMTP

Comptes de service

- Un compte administrateur : L'installation des produits nécessite d'avoir un compte administrateur local sur les serveurs (Compte root)
- Des comptes LDAP dédiés sur l'ITDS Apporteurs :
 - o « PWM Admin » : Avec les droits d'administration sur les utilisateurs (changement de mot de passe, ajout/modification d'attributs, ...)
 - o « Proxy user » : Pour vérifier l'état des utilisateurs → Compte administrateur
- Les utilisateurs doivent avoir le droit de modifier eux-mêmes leur mot de passe dans l'ITDS, comme on est sur du self-service.

2.2. SERVEUR PWM

- Les serveurs nécessaires à l'installation de PWM pour les environnements devront avoir les caractéristiques suivantes :

Prérequis pour le BUILD CARA

- Java 1.8 JDK + → OpenJDK 11 à date

- Maven 3.2 +

Prérequis pour le déploiement de PWM

- Java SE 8 ou supérieur → OpenJDK 11 à date
- Un serveur d'application Java compatible Java Servlet Container v3.0 (Apache Tomcat v7 ou supérieur) → Tomcat 9 à date
- Au moins 512 MB pour la JVM. Environ 1 GB conseillé d'après les volumétries connues.
- 5 GB d'espace disque pour la base locale utilisée pour la configuration par défaut

2.3. RECAPITULATIF DES PREREQUIS TECHNIQUES

Les tableaux suivants listent l'ensemble des éléments nécessaires à l'installation et la configuration de l'environnement cible :

	Nom générique	DTU	Recette	Production
Certificat SSL	<Certificat_Tomcat>	Certificat auto-signés	N/A Certificat SSL porté par la VIP	N/A Certificat SSL porté par la VIP
Compte de service Tomcat	<SVC_Tomcat>	tomcat	tomcat	tomcat
Port d'écoute Tomcat	<PORT_Tomcat>	8443	8080	8080
Compte LDAP sur l'ITDS Apporteurs « PWM Admin »	<PWM_Admin>	uid=svc_pwm_dev,cn=users,c=fr	uid=svc_pwm_rec,cn=users,c=fr	XXX
Compte LDAP sur l'ITDS Apporteurs « PWM Proxy »	<PWM_Proxy>	uid=svc_pwm_dev,cn=users,c=fr	uid=svc_pwm_rec,cn=users,c=fr	XXX
URL VIP PWM	<PWM_URL>	https://dtu-pwdselfcare.swisslife.lan/pwm	https://rec-pwdselfcare.swisslife.lan/pwm	https://pwdselfcare.swisslife.fr/pwm
Serveurs PWM	<PWM_Hostname>	delwww12482 delwww12482	relwww12950 relwww12951	prlwww12948 prlwww12949
PWM Master_Key	<PWM_Master_Key>			
Serveur ITDS Apporteurs	<ITDS_Hostname_B2B>	annuairesITDS-recette.swisslife.lan	annuairesITDS-recette.swisslife.lan	annuairesITDS.swisslife.lan
Port d'écoute de l'ITDS Apporteurs	<ITDS_Port_B2B>	2389	2389	2389
Base de recherche des utilisateurs dans l'ITDS Apporteurs	<ITDS_UserContainer_B2B>	cn=users,C=FR	cn=users,C=FR	cn=users,C=FR
Identifiant utilisateur dans l'ITDS Apporteurs	<ITDS_UID_B2B>	uid	uid	uid
Keystore de la JVM utilisée par Tomcat	<PATH_TO_JAVA_KEYSTORE>	/etc/opt/rh/scl/scl5/tomcat/certificates/delwww12482.swisslife.lan.jks	N/A Certificat SSL porté par la VIP	N/A Certificat SSL porté par la VIP

2.4. PREREQUIS APPLICATIFS

L'application PWM nécessite les prérequis suivants :

Extension de schéma LDAP sur ITDS Apporteurs.

PWM utilise des attributs pour stocker les logs, les questions secrètes ou l'historique des mots de passe.

Il est possible d'utiliser des attributs existants, mais il est conseillé de mettre à jour le schéma LDAP en créant un nouvel objectClass PwmUser et les attributs suivants qui seront ajoutés sur les utilisateurs.

Name	OID	Type
pwmUser	1.3.6.1.4.1.35015.1.1.1	Auxiliary

Name	OID	Type	Single	Description
pwmEventLog	1.3.6.1.4.1.35015.1.2.1	OctetString	N	Logs utilisés par les admin et le helpdesk
pwmResponseSet	1.3.6.1.4.1.35015.1.2.2	OctetString	N	Liste de questions secrètes pour l'oubli de mot de passe
pwmLastPwdUpdate	1.3.6.1.4.1.35015.1.2.3	Time	Y	Dernière mise à jour du mot de passe
pwmGUID	1.3.6.1.4.1.35015.1.2.4	DirectoryString	Y	Identifiant unique utilisé par PWM

Le mapping retenu est le suivant :

Attribut PWM	Attribut ITDS Apporteurs
pwmGUID	ibm-entryuuid
pwmLastPwdUpdate	pwdChangedTime
pwmEventLog	Octet String - pas d'équivalent existant aujourd'hui => Nouvel attribut ITDS
pwmResponseSet	NA
pwmOtpSecret	NA

PWM utilise pour l'administration et le helpdesk des utilisateurs spéciaux et des groupes.

- Les utilisateurs spéciaux sont a minima les suivants :
- Un proxy user pour vérifier l'état des utilisateurs = compte de service
- Des administrateurs pour administrer la configuration PWM
- Des utilisateurs Helpdesk qui pourront vérifier l'état du compte utilisateur et faire un reset du mot de passe

Une phase de test de non régression est à prévoir pour valider la migration de CAS vers PWM.

3. CONFIGURATION DU SERVEUR TOMCAT



Prérequis : Serveur Tomcat installé et configuré en HTTPS selon les standards SwissLife :

- En DTU : SSL de bout en bout (Tomcat et VIP en HTTPS)
- En REC et PROD : SSL Offload avec Tomcat en HTTP/8080 et VIP en HTTPS/443

3.1. INSTALLATION D'OPENJDK 11

L'application nécessite a minima OpenJDK 11 pour fonctionner.

Toutes les actions sont à effectuer avec un compte administrateur local sur le serveur.

1. Installer la dernière version d'OpenJDK

```
yum install java-11-openjdk.x86_64  
update-alternatives --config java
```

3.2. MISE A JOUR DU FICHIER DE CONFIGURATION

PWM nécessite la création d'un répertoire qui sera utilisé pour gérer la configuration propre à l'application. Le serveur doit avoir les droits en lecture/écriture sur ce répertoire.

1. Création du répertoire PWM :

```
mkdir /data/pwm-data  
chmod 755 /data/pwm-data  
chown -R tomcat:tomcat pwm-data
```

2. Ouvrir le fichier de configuration tomcat en édition :

```
vi /etc/opt/rh/scls/jws5/tomcat/tomcat.conf  
OU vi /opt/rh/jws5/root/usr/share/tomcat/conf/tomcat.conf
```

3. Rechercher les lignes suivantes :

```
# If you wish to further customize your tomcat environment,  
# put your own definitions here
```



```
# (i.e. LD_LIBRARY_PATH for some jdbc drivers)
```

4. Ajouter la ligne suivante dans le fichier tomcat.conf :

```
PWM_APPLICATIONPATH=/data/pwm-data
```

3.3. IMPORT DU CERTIFICAT SSL (UNIQUEMENT EN DTU)



- **Sur la Recette et la production, le serveur Tomcat doit être configuré en HTTP sur le port 8080.**

Pour la DTU, les serveurs Tomcat sont en SSL de bout en bout avec la configuration suivante :

1. Ajouter les certificats SSL Swisslife dans le Keystore de la JVM utilisée par Tomcat.
2. Modifier le fichier server.xml pour activer le HTTPS sur le Tomcat.

```
vi /opt/rh/jws5/root/usr/share/tomcat/conf/server.xml
```

3. Ajouter le « Connecteur HTTPS » dans la balise Connector

```
<Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
  maxThreads="150" SSLEnabled="true"
  scheme="https" secure="true"
  keystoreFile="<PATH_TO_JAVA_KEYSTORE>" keystorePass="changeit"
  clientAuth="false" sslProtocol="TLS"
/>
```

Exemple en DTU : /etc/opt/rh/scls/jws5/tomcat/certificates/delwww12482.swisslife.lan.jks (Pour la DTU, utilisation d'un Keystore dédié créé manuellement pour utiliser des certificats SSL auto-signés.

4. Redémarrer le serveur Tomcat :

```
systemctl stop jws5-tomcat.service
systemctl start jws5-tomcat.service
```

4. DEPLOIEMENT DE PWM

Le package de livraison de l'application PWM est composée

- d'un fichier war : **pwm.war**
- d'un fichier de configuration : **PwmConfiguration.xml**

Remarque : Dans le cadre d'une mise à jour, il est recommandé de sauvegarder la configuration actuelle et de l'importer sur la nouvelle version du war (Cf. chapitre [4.3 Backup/Restore de la configuration](#))

4.1. BUILD DEPUIS LE SHAREPOINT

Le build de l'application est disponible dans le répertoire suivant :

https://slfr.sharepoint.com/:f:/r/sites/GRPT_DSI_ID29DeploiementSSOAporteur/Shared%20Documents/PWM/LIVRAISON?csf=1&web=1&e=GIIfNwX

4.2. DETAIL DES VARIABLES DU FICHIER PWMCONFIGURATION.XML

Le tableau suivant présente la liste des variables à modifier en fonction des environnements.

PwmConfiguration	Variable CARA	DTU
pwm.selfURL	%PWM_SELF_URL%	https://dtu-pwdselfcare.swisslife.lan/pwm
email.smtp.type	%EMAIL_SMTP_TYPE%	START_TLS
email.smtp.address	%EMAIL_SMTP_ADDRESS%	smtp.appli.swisslife.fr
email.smtp.userpassword	%EMAIL_SMTP_PASSWORD%	XXX
email.smtp.port	%EMAIL_SMTP_PORT%	587
email.smtp.username	%EMAIL_SMTP_USERNAME%	swissline.ate.rec@swisslife.fr
email.system.from Address	%EMAIL_SYSTEM_FROM_ADDRESS%	ne_pas_repondre@swisslife.fr
ldap.rootContexts	%LDAP_ROOT_CONTEXT%	cn=users,c=fr
ldap.proxy.password	%LDAP_PROXY_PASSWORD%	xxx
ldap.proxy.username	%LDAP_PROXY_USERNAME%	uid=svc_pwm_dev,cn=users,c=fr
ldap.serverUrls	%LDAP_SERVER_URLS%	ldap://annuaireITDS-recette.swisslife.lan:2389

ldap.usernameSearchFilter	%LDAP_USERNAME_SEARCH_FILTER%	(&(objectClass=person)(uid=%USERNAME%)(!(codeAD=*))
ldap.addObjectClasses	%LDAP_ADD_OBJECTCLASSES%	person
ldap.guidAttribute	%LDAP_GUID_ATTRIBUTE%	ibm-entryuuid
ldap.namingAttribute	%LDAP_NAMING_ATTRIBUTE%	uid
passwordLastUpdateAttribute	%LDAP_PASSWORD_LAST_UPDATE_ATTRIBUTE%	pwdChangedTime
pwmAdmin.queryMatch	%PWM_ADMIN_QUERY_MATCH%	{"type":"ldapUser","ldapBase":"uid=svc_pwm_dev,cn=users,C=FR"}
password.allowChange.queryMatch	%PASSWORD_ALLOW_CHANGE_QUERY_MATCH%	{"ldapProfileID":"","ldapQuery":"(&(objectClass=person)(!(codeAD=*))","ldapBase":""}
recovery.queryMatch	%RECOVERY_QUERY_MATCH%	{"ldapQuery":"(&(objectClass=person)(!(codeAD=*))","ldapBase":""}
recovery.searchFilter	%RECOVERY_SEARCH_FILTER%	(&(objectClass=person)(uid=%uid%)(!(codeAD=*))
email.default.fromAddress	%EMAIL_DEFAULT_FROM_ADDRESS%	ne_pas_repondre@swisslife.fr

4.3. DEPLOIEMENT DE L'APPLICATION

Les opérations suivantes doivent être réalisées avec un compte administrateur de domaine.

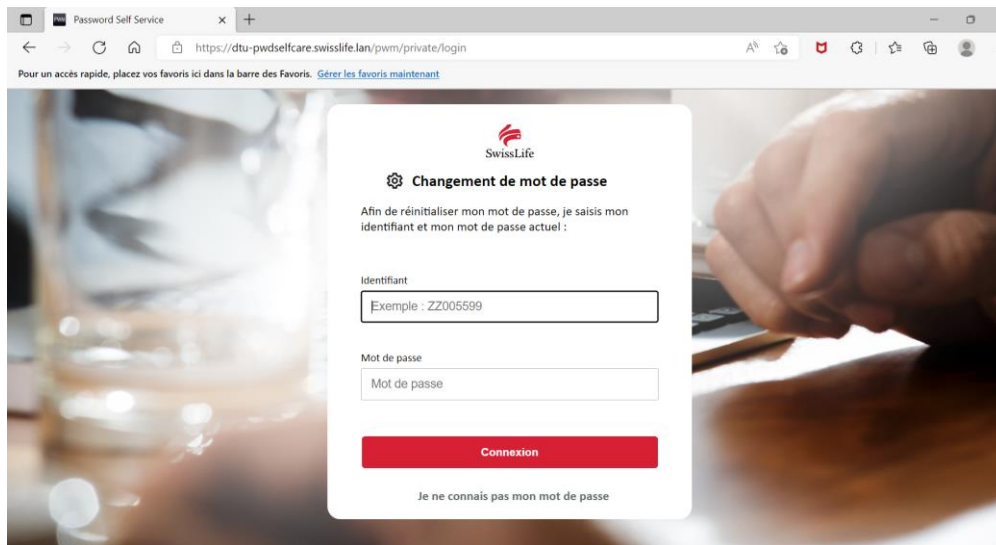
1. Déposer le fichier PwmConfiguration.xml dans le répertoire PWM_APPLICATIONPATH (/data/pwm-data)
2. Déposer le fichier pwm.war dans le répertoire webapps du server Tomcat (/product/tomcat9/webapps/)
3. Ajouter les droits sur les livrables

```
chown tomcat:tomcat PwmConfiguration.xml
chmod 755 pwm.war
```

4. Si nécessaire redémarrer le serveur Tomcat pour forcer le déploiement du war :

```
systemctl stop jws5-tomcat.service
systemctl start jws5-tomcat.service
```

5. Pour vérifier que le déploiement s'est déroulée correctement, vous pouvez vérifier la connexion en ouvrant l'URL suivante depuis le réseau interne et depuis internet : <PWM_URL>
6. L'écran suivant doit s'afficher :



4.4. BACKUP/RESTORE DE LA CONFIGURATION

4.4.1. En ligne de commande

4.4.1.1. Backup de la configuration

1. Se connecter au serveur PWM avec un compte administrateur de domaine.
2. Sauvegarder le fichier PwmConfiguration.xml présent dans le répertoire PWM_APPLICATIONPATH (/data/pwm-data)

```
cp /data/pwm-data/PwmConfiguration.xml /tmp/PwmConfiguration.xml_AAAAMMJJ
```

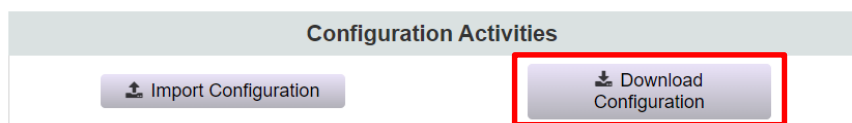
4.4.1.2. Restore de la configuration

1. Se connecter au serveur PWM avec un compte administrateur de domaine.
2. Remplacer le fichier PwmConfiguration.xml présent dans le répertoire PWM_APPLICATIONPATH (/data/pwm-data) par la sauvegarde.

4.4.2. Via l'interface d'administration

4.4.2.1. Backup de la configuration

1. Se connecter à PWM avec un compte administrateur
2. Aller dans Administration > Configuration Manager et se connecter avec le mot de passe Master de l'environnement
3. Cliquer sur le bouton « Download configuration » pour sauvegarder le fichier PwmConfiguration.xml

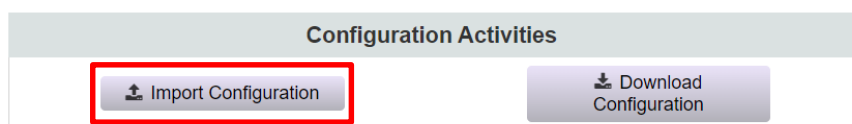


4.4.2.2. Restore de la configuration

Il est possible de restaurer la configuration en déposant le fichier PwmConfiguration.xml dans le répertoire PWM_APPLICATIONPATH (/data/pwm-data).

Sinon, l'interface d'administration permet de restaurer la configuration :

1. Se connecter à PWM avec un compte administrateur
2. Aller dans Administration > Configuration Manager et se connecter avec le mot de passe Master de l'environnement
3. Cliquer sur le bouton « Import configuration » pour sauvegarder le fichier PwmConfiguration.xml



5. DESCRIPTION DES DEVELOPPEMENTS CUSTOM

Les spécificités de l'annuaire ITDS (IBM Tivoli Directory Server) ont nécessités plusieurs ajustements sur le code source de la solution PWM.

5.1. STRUCTURE DU PROJET

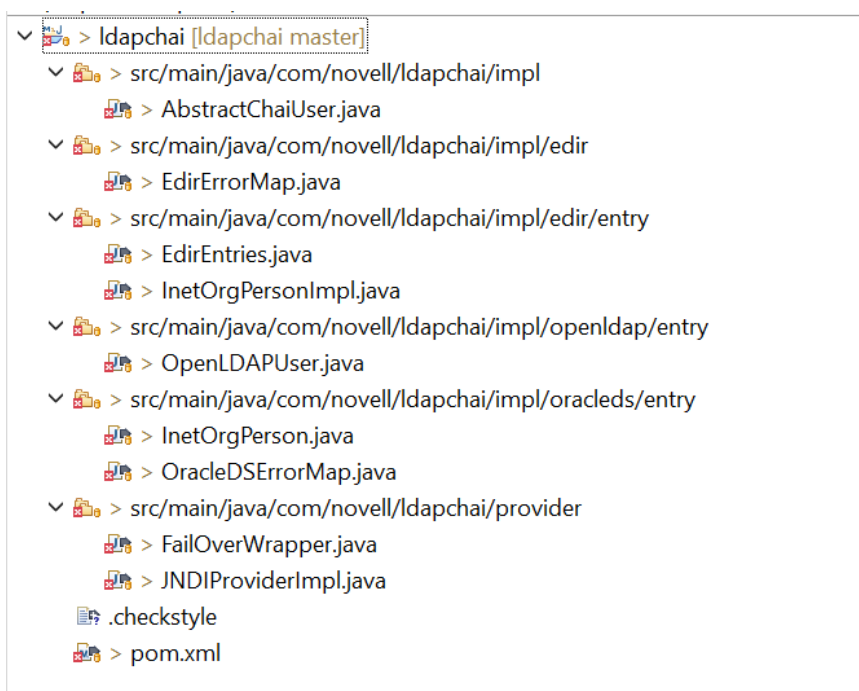
PWM est un projet Java opensource disponible sur <https://github.com/pwm-project/pwm>

L'application est constituée d'un war déployé sur tomcat qui s'appuie sur deux librairies principales :

- pwm-server : contient l'ensemble des fonctions de l'application
- ldapchai : permet de gérer les appels LDAP <https://github.com/ldapchai/ldapchai>

Les chapitres suivants décrivent les différentes modifications réalisées

5.2. LDAPCHAI



5.2.1. AbstractChaiUser

Mauvaise gestion du changement de mot de passe avec ITDS

Dans le cas où le mot de passe existe déjà, il faut appeler la méthode `replaceAttribute`

```
@Override

public void changePassword( final String oldPassword, final String newPassword )

    throws          ChaiUnavailableException,          ChaiPasswordPolicyException,
ChaiOperationException

{
    try
    {
        //202211 - CDE update ITDS Apporteurs pwdSafeModify

        if ( oldPassword == null )

        {
            writeStringAttribute( ATTR_PASSWORD, newPassword );

        }

        else

        {
            replaceAttribute( ATTR_PASSWORD, oldPassword, newPassword );

        }

    }

    catch ( ChaiOperationException e )

    {
        throw ChaiPasswordPolicyException.forErrorMessage( e.getMessage() );

    }

}
```

Gestion des notifications d'expiration de mot de passe.

Valeur par défaut : 90 jours.

```
public Instant readPasswordExpirationDate()
    throws ChaiUnavailableException, ChaiOperationException
{
    //UPDATE CDE

    final      Instant      passwordChangedTime      =      this.readDateAttribute(
ChaiConstant.ATTR_OPENLDAP_PASSWORD_CHANGED_TIME );

    if ( passwordChangedTime != null )
    {
        final String expirationInterval = "7776000";
        if ( expirationInterval != null && expirationInterval.trim().length() > 0 )
        {
            final long explnt = Long.parseLong( expirationInterval ) * 1000L;
            final long pwExpireTimeMs = passwordChangedTime.toEpochMilli() + explnt;
            return Instant.ofEpochMilli( pwExpireTimeMs );
        }
    }

    return null;
}
```

5.2.2. EdirErrorMap

Prise en charge de l'erreur de pwdReset ITDS (le message est différent des autres annuaires)

```
//SWISSLIFE 202210 - CDE - Gestion pwdReset ITDS Apporteurs

PASSWORD_REQ_CHANGES( -953, ChaiError.NEW_PASSWORD_REQUIRED, true,
false,

    "javax.naming.OperationNotSupportedException: [LDAP: error code 53 - Error,
Password must be changed after reset]" ),
```


5.2.3. EdirEntries

ITDS ajoute .Z à la fin de la date standard LDAP (AAAAMMJJHHMMSS)

```
public static Instant convertZuluToInstant( final String input )
{
    if ( input == null )
    {
        throw new NullPointerException();
    }

    try
    {
        //SWISSLIFE - 2021

        final String inputFormatted;

        if ( input.contains( "." ) )
        {
            inputFormatted = input.split( "\\." )[0].concat( "Z" );
        }
        else
        {
            inputFormatted = input;
        }

        final LocalDateTime localDateTime = LocalDateTime.parse( inputFormatted,
            EDIR_TIMESTAMP_FORMATTER );

        final ZonedDateTime zonedDateTime = localDateTime.atZone( ZoneOffset.UTC );

        return Instant.from( zonedDateTime );
    }
    catch ( DateTimeParseException e )
    {
        throw new IllegalArgumentException( "unable to parse zulu time-string: " +
            e.getMessage() );
    }
}
```

```
}  
  
}
```

5.2.4. OracleDSErrorMap

enum OracleDSError

Prise en charge de l'erreur de pwdReset ITDS (le message est différent des autres annuaires)

```
//SWISSLIFE 202210 - CDE - Gestion pwdReset ITDS Apporteurs  
  
    //PASSWORD_REQ_CHANGES( ChaiError.NEW_PASSWORD_REQUIRED, true, false,  
    "error code 53 - Password was reset and must be changed" ),  
  
    PASSWORD_REQ_CHANGES( ChaiError.NEW_PASSWORD_REQUIRED, true, false,  
        "javax.naming.OperationNotSupportedException: [LDAP: error code 53 - Error,  
    Password must be changed after reset]" ),
```

5.2.5. FailOverWrapper

Ajout d'une exception si le mot de passe doit être changé après le pwdReset

```
private Object failOverInvoke( final Method m, final Object[] args )  
  
    throws ChaiException  
  
catch ( Exception e )  
  
    {  
  
        //SWISSLIFE 202210 - CDE - Gestion pwdReset ITDS Apporteurs  
  
        if ( e.getMessage().contains( "[LDAP: error code 53 - Error, Password must be changed  
after reset]" ) )  
  
            {  
  
                throw new ChaiUnavailableException( "ITDS PwdReset - [LDAP: error code 53 -  
Error, Password must be changed after reset]", ChaiError.NEW_PASSWORD_REQUIRED );  
  
            } //FIN CDE
```

```

        if ( settings.errorIsRetryable( e ) && !closed )
        {
            rotationMachine.reportBrokenProvider( currentProvider, e );
        }
        else
        {
            throw AbstractProvider.convertInvocationExceptionToChaiException( e );
        }
    }
}

```

5.2.6. JNDIProviderImpl

Gestion des attributs multivalué pour la notification d'expiration de mot de passe avec pwmData

```

public class JNDIProviderImpl extends AbstractProvider implements ChaiProviderImplementor

@SuppressFBWarnings( "DCN_NULLPOINTER_EXCEPTION" )

    public final Set<String> readMultiStringAttribute( final String entryDN, final String
attributeName )

        throws ChaiUnavailableException, ChaiOperationException
    {
        activityPreCheck();

        getInputValidator().readMultiStringAttribute( entryDN, attributeName );

        final Set<String> attributeValues = new HashSet<>();

        NamingEnumeration namingEnum = null;

        try
        {

            // Get only the Attribute that is passed in.

```

```

final String[] attributesArray = {attributeName};

// Get the Enumeration of attribute values.

final LdapContext ldapConnection = getLdapConnection();

//Begin CDE */

// Get only the Attribute that is passed in.

final Attributes returnedAttribs;

        returnedAttribs    =    ldapConnection.getAttributes(    addJndiEscape(    entryDN    ),
attributesArray );

final NamingEnumeration attrEnumeration = returnedAttribs.getAll();

while ( attrEnumeration.hasMoreElements() )
{
    final Attribute attribute = ( Attribute ) attrEnumeration.nextElement();

    if ( attribute != null )
    {
        namingEnum = attribute.getAll();

        while ( namingEnum.hasMoreElements() )
        {
            final Object attributeValue = namingEnum.nextElement();

            if ( attributeValue instanceof String )
            {
                attributeValues.add( (String) attributeValue );
            }

            else
            {
                String stringValue = "";

```

```

        final byte[] buf = ( byte[] ) attributeValue;

        for ( int i = 0; i < buf.length; i++ )
        {
            stringValue = stringValue.concat( Integer.toHexString( buf[i] ) );
        }

        stringValue = new String( buf, StandardCharsets.UTF_8 );

        attributeValues.add( stringValue );
    }
}

}

}

//end CDE

/*

```

5.2.7. checkstyle

Modification du checkstyle pour pouvoir compiler

```

<?xml version="1.0" encoding="UTF-8"?>

<fileset-config file-format-version="1.2.0" simple-config="false" sync-formatter="false">

    <local-check-config                name="maven-checkstyle-plugin"                validate
location="file:/C:/Users/cderval.SWISSLIFE/git/ldapchai/src/build/checkstyle.xml" type="remote"
description="maven-checkstyle-plugin configuration validate">

        <property name="basedir" value="C:\Users\cderval.SWISSLIFE\git\ldapchai"/>

        <property name="checkstyle.header.file" value="C:\Users\cderval.SWISSLIFE\eclipse-
workspace\.metadata\plugins\org.eclipse.core.resources\projects\ldapchai\com.basistech.m2e.
code.quality.checkstyleConfigurator\checkstyle-header-validate.txt"/>

        <property name="checkstyle.cache.file" value="${project_loc}/target/checkstyle-cachefile"/>

    </local-check-config>

    <fileset name="java-sources-validate" enabled="true" check-config-name="maven-checkstyle-
plugin validate" local="true">

        <file-match-pattern match-pattern="^src/main/resources.*\properties" include-pattern="true"/>
    </fileset>
</fileset-config>

```


























```
<file-match-pattern match-pattern="^src/main/java/.*\\.properties" include-pattern="true"/>
</fileset>
</fileset-config>
```

5.2.8. Pom.xml

Création d'une version de librairie propre à Swisslife pour la compatibilité ITDS.

```
<version>0.8.3-SWISSLIFE</version>
```

5.3. PWM-SERVER

- ✓  > pwm-parent [pwm master]
 - ✓  > build
 -  > checkstyle-suppression.xml
 - ✓  > server
 -  > pom.xml
 - ✓  > server/src/main/java/password/pwm/http/servlet/changepw
 -  > ChangePasswordServlet.java
 -  > ChangePasswordServletUtil.java
 - ✓  > server/src/main/java/password/pwm/http/servlet/forgottenpw
 -  > ForgottenPasswordServlet.java
 - ✓  > server/src/main/java/password/pwm/ldap/auth
 -  > LDAPAuthenticationRequest.java
 - ✓  > server/src/main/java/password/pwm/util/password
 -  > PasswordUtility.java
 - ✓  > webapp/src/main/webapp/public/resources/themes/swisslife
 -  background-connection.jpg
 -  background-pwm.png
 -  logo-petit-sl.svg
 -  logo-SwissLife.jpg
 -  logoSwissLife.png
 -  mobileStyle.css
 -  pwm-check.svg
 -  pwm-roue.svg
 -  style.css
 -  > pom.xml

5.3.1. checkstyle-suppression.xml

```
<!-- Suppression du check sur les images du thème Swisslife -->
<suppress files="logo-petit-sl.svg" checks="[a-zA-Z0-9]*/>
<suppress files="pwm-roue.svg" checks="[a-zA-Z0-9]*/>
<suppress files="pwm-check.svg" checks="[a-zA-Z0-9]*/>
```

5.3.2. Pom.xml

Remplacement de la dépendance LDAPCHAI pour utiliser la version ITDS.

```
<dependency>
    <groupId>com.github.ldapchai</groupId>
    <artifactId>ldapchai</artifactId>
    <version>0.8.3-SWISSLIFE</version>
    <exclusions>
```

5.3.3. ChangePasswordServlet

Gestion des champs anciens et nouveaux mot de passe

```
@ExceptionHandler( action = "change" )

public ProcessStatus processChangeAction( final PwmRequest pwmRequest ) throws
ServletException, PwmUnrecoverableException, IOException, ChaiUnavailableException
{
    final ChangePasswordBean changePasswordBean = getBean( pwmRequest );

    final UserInfo userInfo = pwmRequest.getPwmSession().getUserInfo();
```

```

        if ( !changePasswordBean.isAllChecksPassed() )
        {
            return ProcessStatus.Continue;
        }

        // 202211 - Ajout CDE - current/oldPassword field

        //if public = pas de oldPassword ; if private : old Password

        boolean isForgotPassword = false;

        PasswordData currentPassword = null;

        if (      pwmRequest.getPwmSession().getLoginInfoBean().getAuthFlags().contains(
AuthenticationType.AUTH_FROM_PUBLIC_MODULE ) )
        {
            isForgotPassword = true;
        }

        if (      pwmRequest.getPwmSession().getLoginInfoBean().getType()      ==
AuthenticationType.AUTH_FROM_PUBLIC_MODULE )
        {
            isForgotPassword = true;
        }

        if ( !isForgotPassword )
        {
            currentPassword = pwmRequest.readParameterAsPassword( "currentPassword" )

                .orElseThrow(      ()      ->      PwmUnrecoverableException.newException(
PwmError.ERROR_FIELD_REQUIRED, "missing currentPassword field" ) );
        }

        // FIN CDE

        final PasswordData password1 = pwmRequest.readParameterAsPassword( "password1" )

            .orElseThrow(      ()      ->      PwmUnrecoverableException.newException(
PwmError.ERROR_FIELD_REQUIRED, "missing password1 field" ) );

        final PasswordData password2 = pwmRequest.readParameterAsPassword( "password2" )

```



```

        .orElseThrow( () -> PwmUnrecoverableException.newException(
PwmError.ERROR_FIELD_REQUIRED, "missing password2 field" ) );

        // check the password meets the requirements

        try

        {

            final ChaiUser theUser = pwmRequest.getClientConnectionHolder().getActor( );

            final PwmPasswordRuleValidator pwmPasswordRuleValidator =
PwmPasswordRuleValidator.create(
                pwmRequest.getLabel(), pwmRequest.getPwmDomain(),
                userInfo.getPasswordPolicy() );

            final PasswordData oldPassword =
pwmRequest.getPwmSession().getLoginInfoBean().getUserCurrentPassword();

            pwmPasswordRuleValidator.testPassword( pwmRequest.getLabel(), password1,
oldPassword, userInfo, theUser );

        }

        catch ( final PwmDataValidationException e )

        {

            setLastError( pwmRequest, e.getErrorInformation() );

            LOGGER.debug( pwmRequest, () -> "failed password validation check: " +
e.getErrorInformation().toDebugStr() );

            return ProcessStatus.Continue;

        }

        //make sure the two passwords match

        final boolean caseSensitive =
userInfo.getPasswordPolicy().getRuleHelper().readBooleanValue(
            PwmPasswordRule.CaseSensitive );

        if ( PasswordUtility.PasswordCheckInfo.MatchStatus.MATCH !=
PasswordUtility.figureMatchStatus( caseSensitive,
            password1, password2 ) )

        {

            setLastError( pwmRequest, PwmError.PASSWORD_DOESNOTMATCH.toInfo() );

```

```

        forwardToChangePage( pwmRequest );

        return ProcessStatus.Continue;

    }

    try
    {
        // 202211 - Ajout CDE - current/oldPassword field nouvelle fonction

        //if public = pas de oldPassword ; if privé : old Password

        if ( isForgotPassword )
        {
            ChangePasswordServletUtil.executeChangePassword( pwmRequest, password1 );
        }

        else
        {
            LOGGER.debug( pwmRequest, () -> "executeChangePassword( pwmRequest,
currentPassword, password1 )" );

            ChangePasswordServletUtil.executeChangePassword(                pwmRequest,
currentPassword, password1 );

        }
    }

    catch ( final PwmOperationalException e )
    {
        LOGGER.debug( () -> e.getErrorInformation().toDebugStr() );

        setLastError( pwmRequest, e.getErrorInformation() );

    }

    return ProcessStatus.Continue;

}

```

5.3.4. ChangePasswordServletUtil

```
//202211 - Ajout CDE nouvelle fonction current + new Password

static void executeChangePassword(

    final PwmRequest pwmRequest,

    final PasswordData currentPassword,

    final PasswordData newPassword

)

    throws          ChaiUnavailableException,          PwmUnrecoverableException,
PwmOperationalException

{

    final PwmDomain pwmDomain = pwmRequest.getPwmDomain();

    final PwmSession pwmSession = pwmRequest.getPwmSession();

    // password accepted, setup change password

    final ChangePasswordBean cpb = pwmDomain.getSessionStateService().getBean(
pwmRequest, ChangePasswordBean.class );

    // change password

    PasswordUtility.setActorPassword(    pwmRequest,    pwmDomain,    currentPassword,
newPassword );

    //init values for progress screen

    {

        final PasswordChangeProgressChecker.ProgressTracker tracker = new
PasswordChangeProgressChecker.ProgressTracker();

        final PasswordChangeProgressChecker checker = new
PasswordChangeProgressChecker(

            pwmDomain,

            ChangePasswordServlet.getProfile( pwmRequest ),

            pwmRequest.getUserInfoIfLoggedIn(),

            pwmRequest.getLabel(),

            pwmRequest.getLocale()


```

```

    );

    cpb.setChangeProgressTracker( tracker );

    cpb.setChangePasswordMaxCompletion( checker.maxCompletionTime( tracker ) );
}

// send user an email confirmation

ChangePasswordServletUtil.sendChangePasswordEmailNotice( pwmRequest );

// send audit event

AuditServiceClient.submitUserEvent( pwmRequest, AuditEvent.CHANGE_PASSWORD,
pwmSession.getUserInfo() );
}

```

5.3.5. ForgottenPasswordServlet

Problème sur les webmails : passe en anglais et boucle sur les liens reçus

```

public ProcessStatus preProcessCheck( final PwmRequest pwmRequest ) throws
PwmUnrecoverableException, IOException, ServletException

    //CDE - 20230317

    //Disable change of locale - bug with webmail when reload "en" to "fr"

    //checkForLocaleSwitch( pwmRequest, forgottenPasswordBean );

```

5.3.6. LDAPAuthenticationRequest

Gestion pwdReset ITDS Apporteurs

```

private AuthenticationResult authenticateUserImpl(

```

```
//SWISSLIFE 202210 - CDE - Gestion pwdReset ITDS Apporteurs

    else

    {

        log( PwmLogLevel.DEBUG,

            () -> "ITDS Apporteurs - auth bind failed, but will allow login due to 'pwdReset'
user attribute', error: "

                + e.getErrorInformation().toDebugStr() );

        allowBindAsUser = true;

        permitAuthDespiteError = true;

    } //FIN CDE
```

Gestion pwdReset ITDS Apporteurs

```
private void testCredentials(

    final UserIdentity userIdentity,

    final PasswordData password

)

    throws PwmUnrecoverableException, PwmOperationalException

// SWISSLIFE 202210 - CDE - Gestion pwdReset ITDS Apporteurs

    if ( e.getErrorCode() != null && e.getErrorCode() ==
ChaiError.NEW_PASSWORD_REQUIRED )

    {

        if ( e.getMessage().contains( "[LDAP: error code 53 - Error, Password must be changed
after reset]" ) )

        {

            final String errorMsg = "Forgot Password attempt for user " + userIdentity + " ByPass
ITDS Apporteurs PwdReset : " + e.getMessage();

            final ErrorInformation errorInformation = new ErrorInformation(
PwmError.PASSWORD_NEW_PASSWORD_REQUIRED, errorMsg );
```

```

        log( PwmLogLevel.WARN, errorInformation::toDebugStr );

        bindSucceeded = true;

        throw new PwmOperationalException( errorInformation );

    }

} //FIN CDE

```

5.3.7. PasswordUtility

Gestion old password ITDS Apporteurs

```

public static void setActorPassword(

    final PwmRequest pwmRequest,

    final PwmDomain pwmDomain,

    final PasswordData newPassword

)

boolean setPasswordWithoutOld = false;

if ( oldPassword == null )

{

    //SWISSLIFE 202211 - CDE - Gestion old password ITDS Apporteurs

    //if ( ( pwmRequest.getClientConnectionHolder().getActor(
    ).getChaiProvider().getDirectoryVendor() == DirectoryVendor.ACTIVE_DIRECTORY )

        final DirectoryVendor currentDirectoryVendor =
pwmRequest.getClientConnectionHolder().getActor( ).getChaiProvider().getDirectoryVendor();

        if ( currentDirectoryVendor == DirectoryVendor.ACTIVE_DIRECTORY ||
currentDirectoryVendor == DirectoryVendor.GENERIC )

        {

            setPasswordWithoutOld = true;

        }

    }

}

```

```
}
```

```
//202211 - Ajout CDE ajout fonction current + new Password

public static void setActorPassword(
    final PwmRequest pwmRequest,
    final PwmDomain pwmDomain,
    final PasswordData currentPassword,
    final PasswordData newPassword
)
    throws ChaiUnavailableException, PwmUnrecoverableException,
    PwmOperationalException
{
    final PwmSession pwmSession = pwmRequest.getPwmSession();
    final UserInfo userInfo = pwmSession.getUserInfo();

    if ( !pwmRequest.getDomainConfig().readSettingAsBoolean(
        PwmSetting.CHANGE_PASSWORD_ENABLE ) )
    {
        final String errorMsg = "attempt to setActorPassword, but user does not have password
        change permission";

        final ErrorInformation errorInformation = new ErrorInformation(
            PwmError.ERROR_UNAUTHORIZED, errorMsg );

        throw new PwmOperationalException( errorInformation );
    }

    if ( pwmRequest.getChangePasswordProfile() == null )
    {
        final String errorMsg = "attempt to setActorPassword, but user does not have change
        password profile assigned";
    }
}
```

```

        final ErrorInformation errorInformation = new ErrorInformation(
PwmError.ERROR_UNAUTHORIZED, errorMsg );

        throw new PwmOperationalException( errorInformation );
    }

    // double check to make sure password meets PWM rule requirements. This should
    // have been done before setActorPassword() is invoked, so it should be redundant
    // but we do it just in case.

    try
    {
        final PwmPasswordRuleValidator pwmPasswordRuleValidator =
PwmPasswordRuleValidator.create( pwmRequest.getLabel(), pwmDomain,
userInfo.getPasswordPolicy() );

        pwmPasswordRuleValidator.testPassword( pwmSession.getLabel(), newPassword, null,
userInfo, pwmRequest.getClientConnectionHolder().getActor() );
    }

    catch ( final PwmDataValidationException e )
    {
        final String errorMsg = "attempt to setActorPassword, but password does not pass local
policy validator";

        final ErrorInformation errorInformation = new ErrorInformation(
e.getErrorInformation().getError(), errorMsg );

        throw new PwmOperationalException( errorInformation );
    }

    final ChaiProvider provider =
pwmRequest.getClientConnectionHolder().getActorChaiProvider();

    setPassword( pwmDomain, pwmRequest.getLabel(), provider, userInfo, currentPassword,
newPassword );

    // update the session state bean's password modified flag

    pwmSession.getSessionStateBean().setPasswordModified( true );

```



```
// update the login info bean with the user's new password

pwmSession.getLoginInfoBean().setUserCurrentPassword( newPassword );

//close any outstanding ldap connections (since they cache the old password)

pwmRequest.getClientConnectionHolder().updateUserLdapPassword(
userInfo.getUserIdentity(), newPassword );

// clear the "requires new password flag"

pwmSession.getLoginInfoBean().getLoginFlags().remove(
LoginInfoBean.LoginFlag.forcePwChange );

// mark the auth type as authenticatePd now that we have the user's natural password.

pwmSession.getLoginInfoBean().setType( AuthenticationType.AUTHENTICATED );

// update the uibean's "password expired flag".

pwmSession.reloadUserInfoBean( pwmRequest );

// create a proxy user object for pwm to update/read the user.

final ChaiUser proxiedUser = pwmRequest.getClientConnectionHolder().getActor();

// update statistics

{

    StatisticsClient.incrementStat( pwmRequest, Statistic.PASSWORD_CHANGES );

}

{

    // execute configured actions

    LOGGER.debug( pwmRequest, () -> "executing configured actions to user " +
proxiedUser.getEntryDN() );
}
```

```

        final      ChangePasswordProfile      changePasswordProfile      =
pwmRequest.getChangePasswordProfile();

        final      List<ActionConfiguration>      actionConfigurations      =
changePasswordProfile.readSettingAsAction(
PwmSetting.CHANGE_PASSWORD_WRITE_ATTRIBUTES );

        if ( !CollectionUtil.isEmpty( actionConfigurations ) )

        {

            final LoginInfoBean clonedLoginInfoBean = JsonFactory.get().cloneUsingJson(
pwmSession.getLoginInfoBean(), LoginInfoBean.class );

            clonedLoginInfoBean.setUserCurrentPassword( newPassword );


            final MacroRequest macroRequest = MacroRequest.forUser(

                pwmDomain.getPwmApplication(),

                pwmRequest.getLabel(),

                pwmSession.getUserInfo(),

                clonedLoginInfoBean

            );


            final ActionExecutor actionExecutor = new ActionExecutor.ActionExecutorSettings(
pwmDomain, userInfo.getUserIdentity() )

                .setMacroMachine( macroRequest )

                .setExpandPwmMacros( true )

                .createActionExecutor();

            actionExecutor.executeActions( actionConfigurations, pwmRequest.getLabel() );

        }

    }

    // invoke post password change actions

    invokePostChangePasswordActions( pwmRequest );


    //update the current last password update field in ldap

```

```
LdapOperationsHelper.updateLastPasswordUpdateAttribute(  
    pwmRequest.getLabel(), userInfo.getUserIdentity() );  
    pwmDomain,  
    }  
}
```

5.4. PWM WEBAPP

Ajout du thème Swisslife dans Webapp