Université IBM i

19 et 20 novembre 2024

IBM Innovation Studio Paris

uii2024
#ibmi
#uii2024

S18 – Services Web avec IWS : sécurité avancée

19 novembre 14:45 - 15:45

Nathanaël Bonnet
GAIA / VOLUBIS
nathanael.bonnet@gaia.fr

IBM

common FRANCE

Université IBM i

19 et 20 novembre 2024

IBM i
continuous innovation
continuous integration

Nathanaël BONNET

IBM i depuis 1999

Expert IBM i

IBM Champion 5 Year Milestone
Awarded through the program
IBM Champion

IBM Professional Certified Developer IBM i 7.x

IBM i Community Advocate Level 3
IBM Power Systems

common FRANCE

MEMBER OF common EUROPE

IBM

Université IBM i

19 et 20 novembre 2024

IBM i
continuous innovation
continuous integration

GAIA
VOTRE EXPERT IBM i

Centre de services IBM i
- Mutualisé
- A distance

Modernisation
- Accompagnement
- Prototypage

Expertise technique
- Prestations de service (audit, consulting…)
- Transfert de connaissances (workshops)

VOLUBIS
DEVENEZ UN EXPERT

FORMATIONS
SUR SITE
A DISTANCE
INTER
SUR MESURE

COURS EN LIGNE
FORMATS COURS
REPLAYS EN LIGNE

BASE DE CONNAISSANCE
IBM i
EN LIGNE

IBM

IBM i
continuous innovation
continuous integration

GAIA
VOTRE EXPERT IBM i

C'est aussi des produits

GAIA

GRMT5250

Consultez l'écran 5250 d'un autre utilisateur en temps réel directement sur l'IBM i

PRINCIPE

GRMT5250 vous permet d'accéder rapidement à l'écran 5250 d'un autre utilisateur, directement depuis votre session IBM i.
Il ne nécessite pas de passer par des outils intermédiaires tel que Teams.

ABONNEMENT ANNUEL 2000€ € HT / PARTITION

REPRENEZ LE CONTRÔLE

Inspiré de notre propre expérience dans la maintenance sur IBM i, nous l'avons conçu pour les professionnels de l'IBM i. Cette solution est idéale pour intervenir rapidement sur les sessions utilisateurs bloquées ou pour diagnostiquer et corriger des bugs en temps réel.
En plus de cette fonctionnalité clé, notre outil offre une gamme d'options supplémentaires qui simplifient la gestion des sessions utilisateurs sur l'IBM i, améliorant ainsi l'efficacité et la réactivité de votre équipe technique. Optez pour une solution intuitive et intégrée, qui vous permet de superviser au mieux les travaux de votre IBM i depuis votre poste de travail.

PRINCIPALES CARACTÉRISTIQUES

- Piloter simplement les travaux 5250
- Consulter les activités utilisateurs en temps réel
- Tous vos travaux interactifs et batch regroupés au même endroit via la console de GRMT5250
- Intervenir rapidement
- Gérer les messages en interrogation sur les sessions utilisateurs
- Gérer les sessions inactives

CONTACT

Demandez votre démo !

04 72 53 00 12
contact@gaia.fr
https://www.gaia.fr

IBM

Services Web avec IWS : sécurité avancée

- Agenda

  - Rappels bonnes pratiques
  - JWT
  - Génération de jetons
  - Validation de jetons

Université IBM i

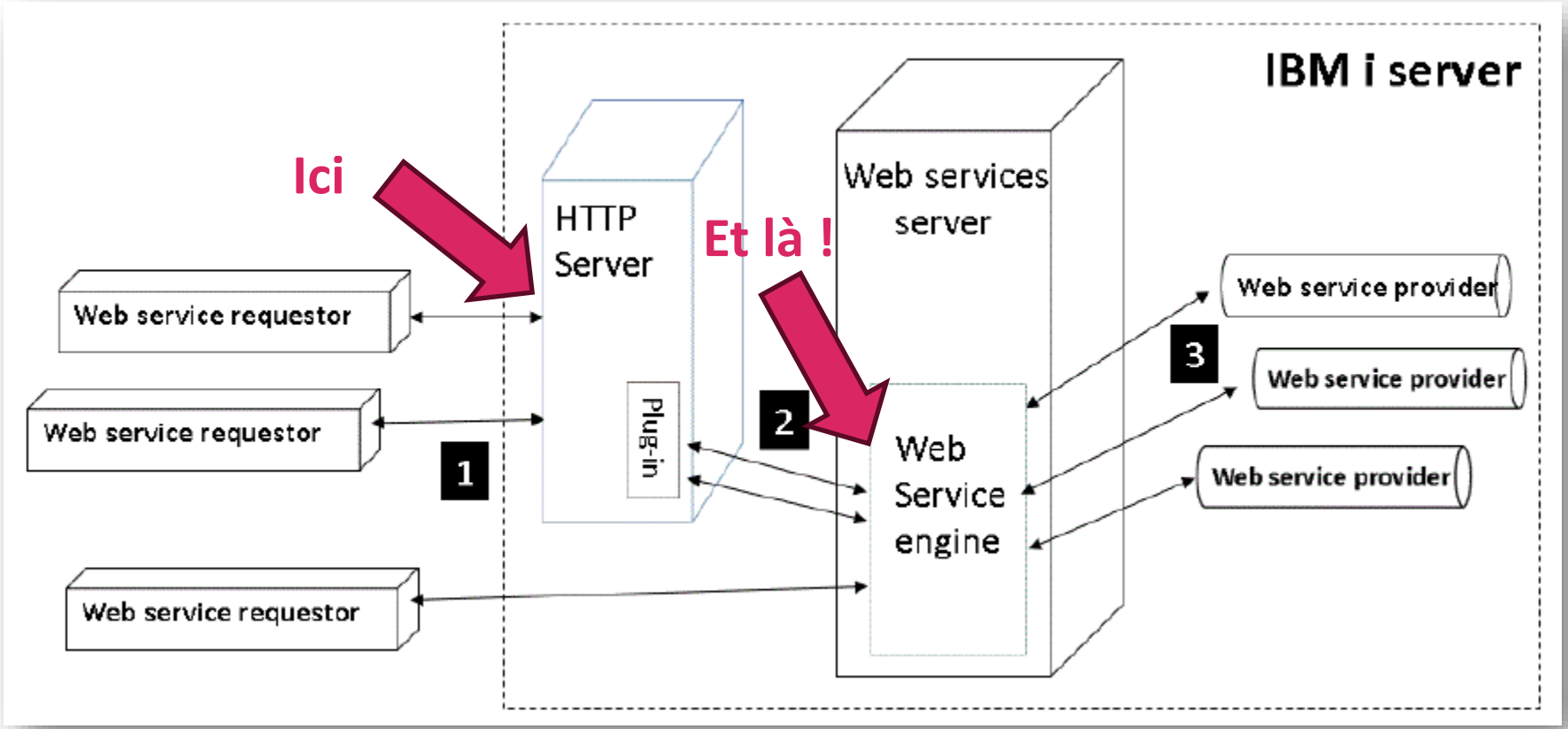
19 et 20 novembre 2024

IBM i
continuous innovation
continuous integration

IBM

Bonnes pratiques

Certificat

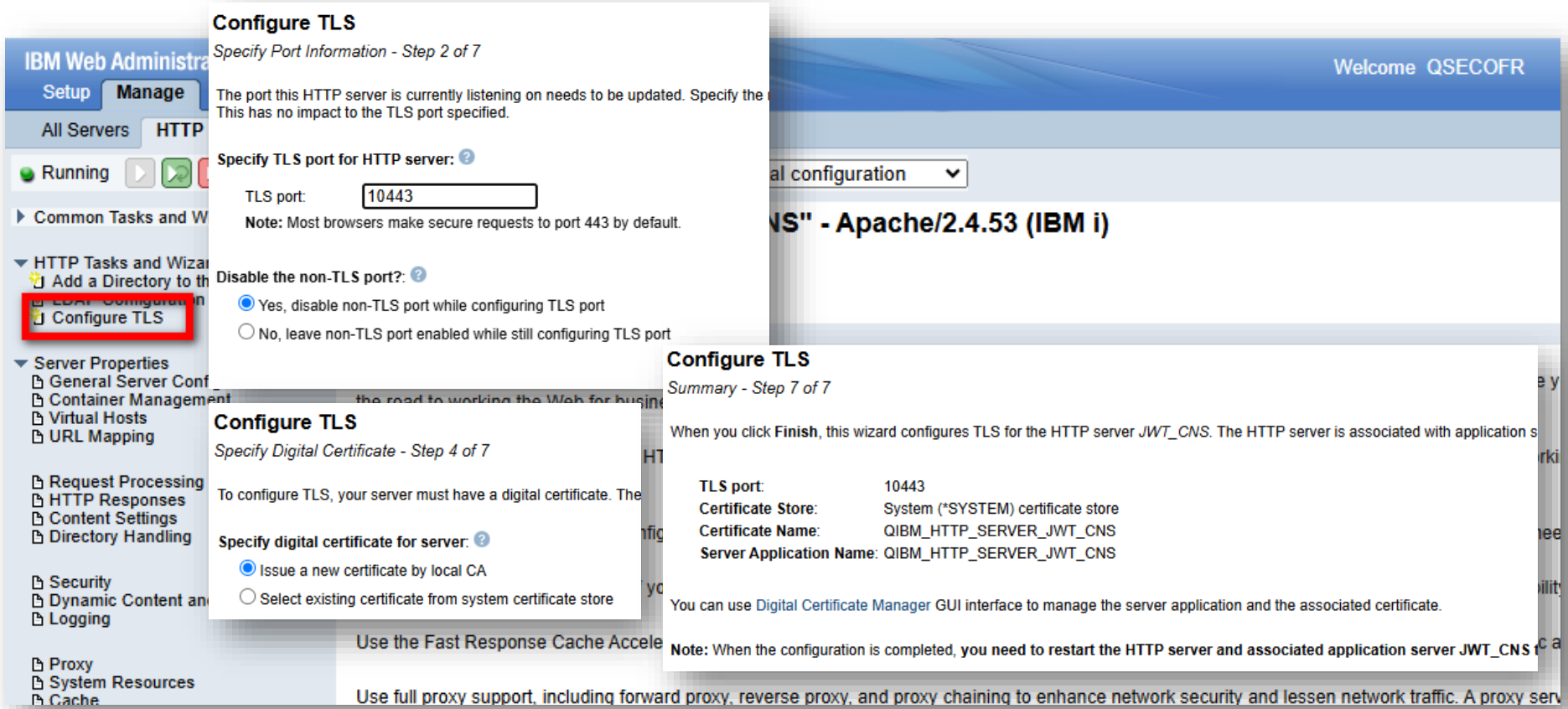

IBM i server
Ici
Et là !
HTTP Server
Plug-in
Web services server
Web Service engine
Web service requestor
Web service requestor
Web service requestor
1
2
3
Web service provider
Web service provider
Web service provider

Certificat

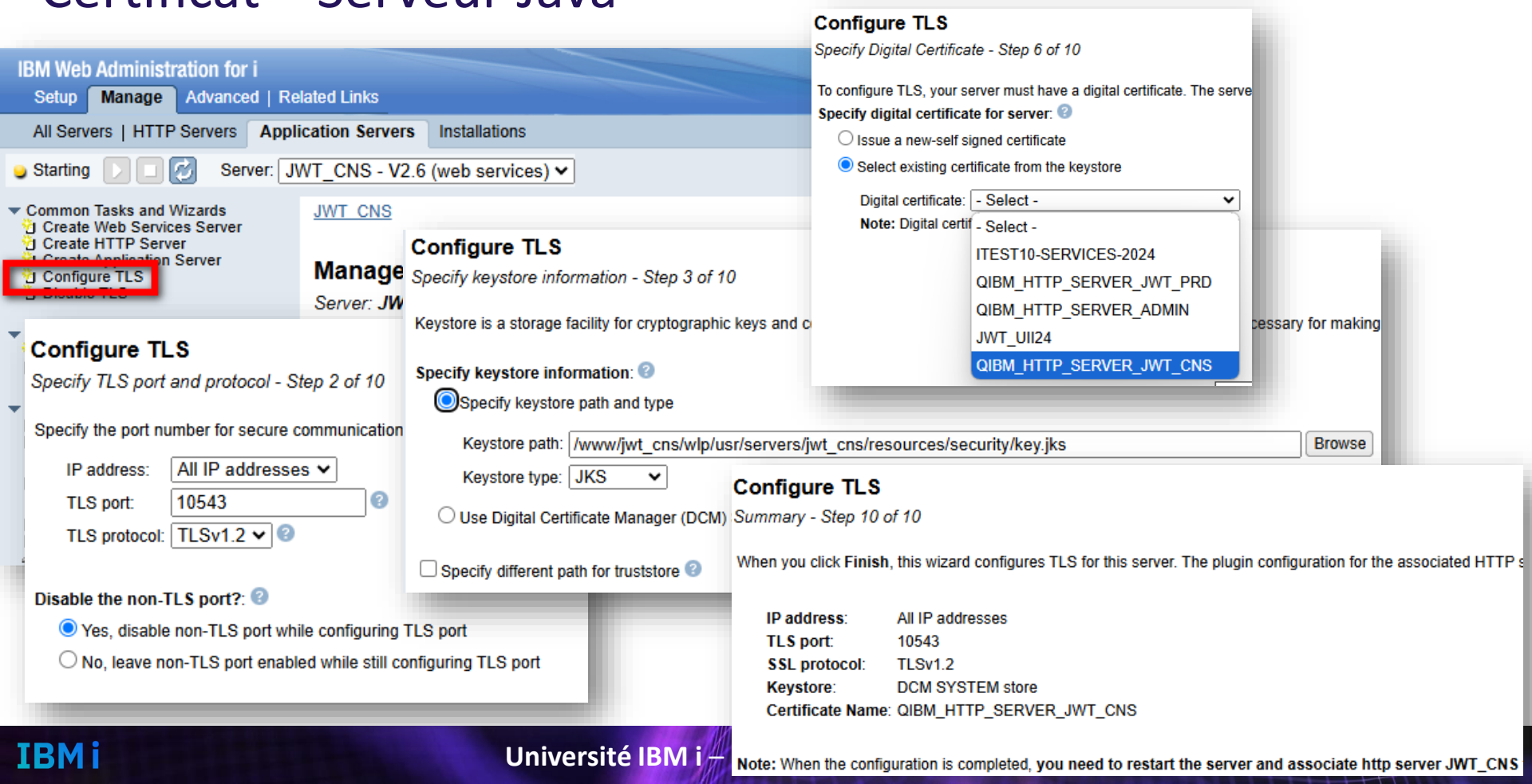
- Ce peut être le même certificat
  - Ou non
  - Dépend de vos besoins (certification en relation avec le nom DNS …)

- Pour les 2 serveurs, des assistants dans l'interface graphique
  - http://votre_partition:2001/HTTPAdmin ou https://votre_partition:2010/HTTPAdmin

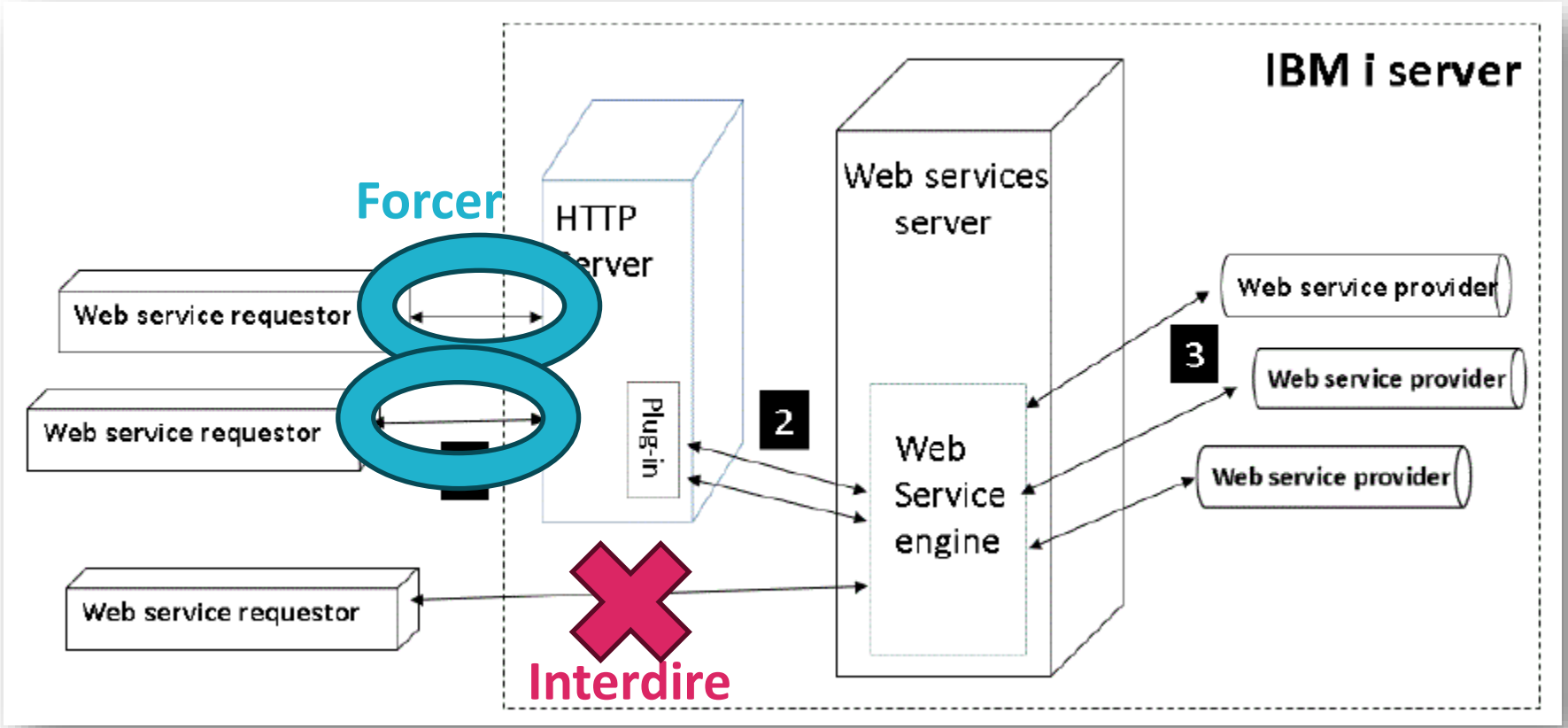
Certificat – Serveur Apache



Certificat – Serveur Java



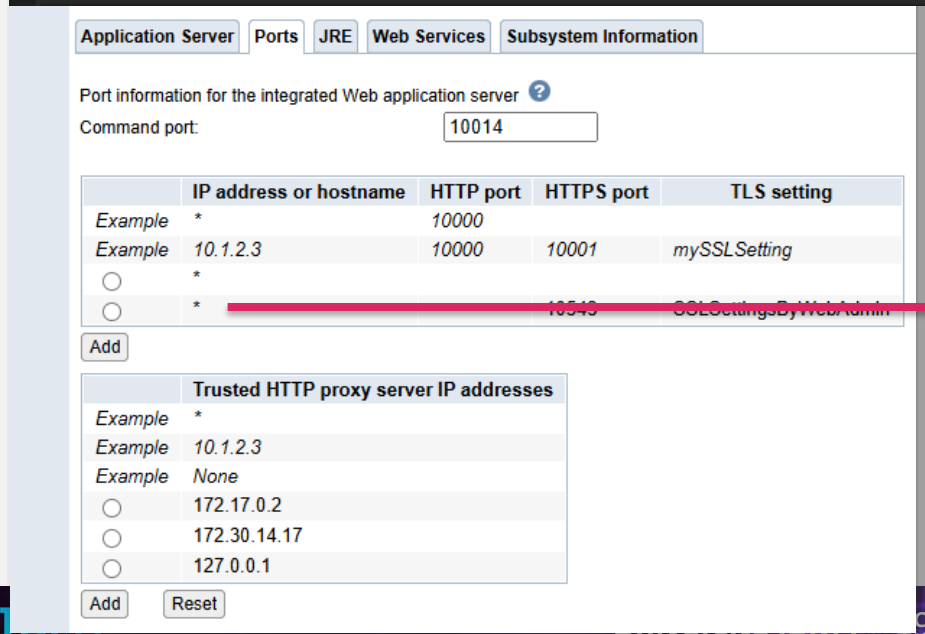
Endpoint

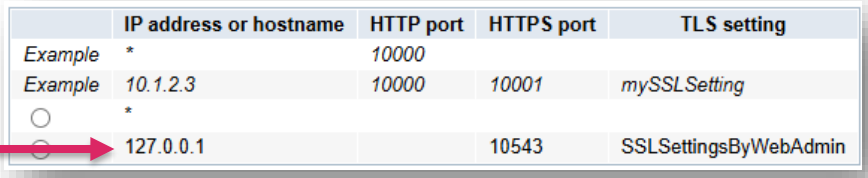

IBM i server
Forcer
HTTP
Web services
server
Web service requestor
Web service requestor
Web service requestor
Plug-in
2
Web
Service
engine
3
Web service provider
Web service provider
Web service provider
Interdire

Endpoint

- Fichier \www*{instance}*\wlp\usr\servers*{instance}*\server.xml

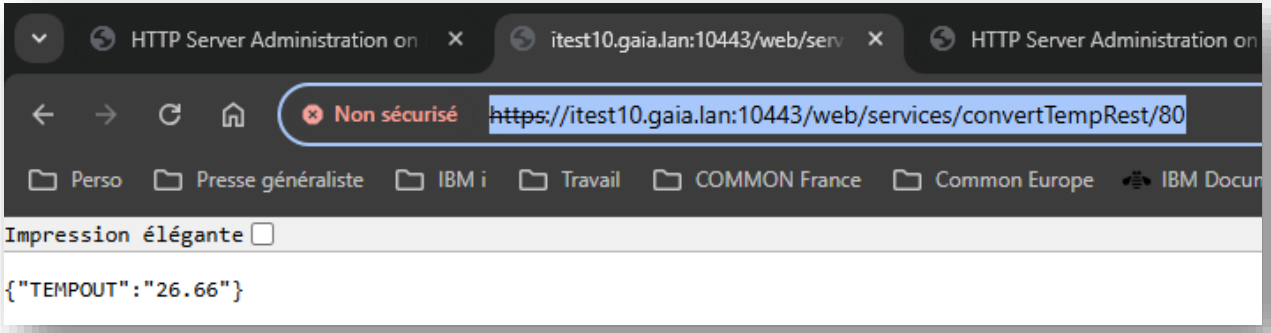
```
<httpEndpoint accessLoggingRef="defaultAccessLogging" host="*" httpOptionsRef="defaultHttpOptions" httpPor
<httpEndpoint accessLoggingRef="defaultAccessLogging" host="127.0.0.1" httpPort="-1" httpsPort="10543" id="
  <sslOptions sslRef="SSLSettingsByWebAdmin"/>
</httpEndpoint>
```



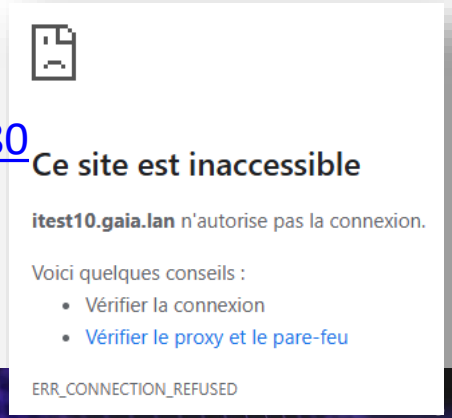


Endpoint

- Par le serveur APACHE
  - https://itest10.gaia.lan:10443/web/services/convertTempRest/80



- Par le serveur JAVA
  - https://itest10.gaia.lan:10543/web/services/convertTempRest/80

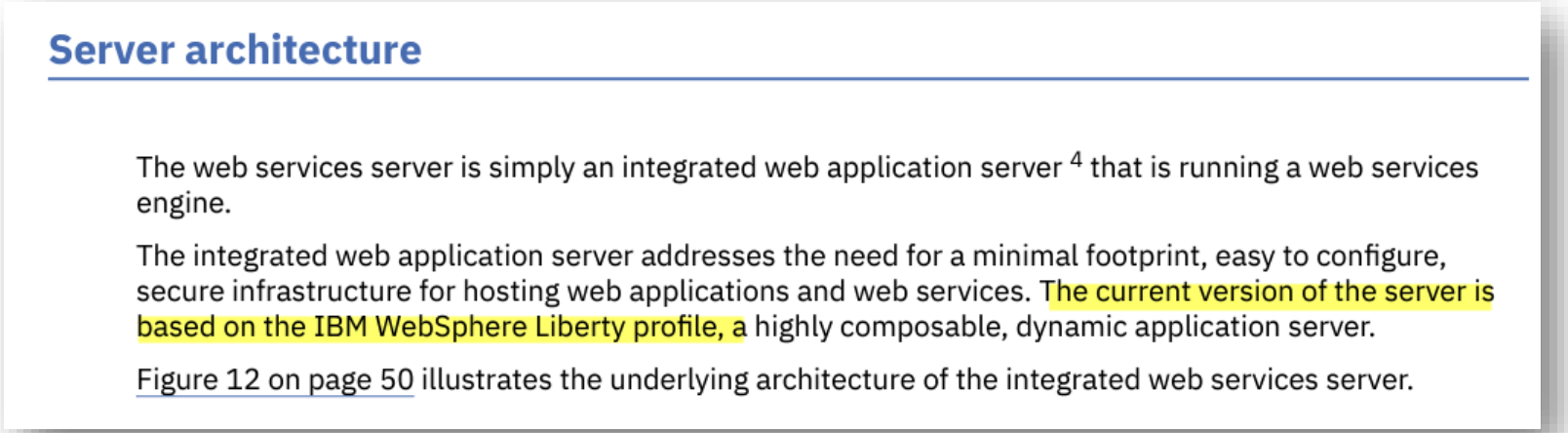


IBM i
continuous innovation
continuous integration

JWT

Liberty ?

- IWS, le serveur de web service de l'IBM i
  - Est un serveur Liberty, « bridé » pour ne faire que du web service
    - https://public.dhe.ibm.com/systems/support/i/iws/systems_i_software_iws_pdf_WebServicesServer_new.pdf



Server architecture

The web services server is simply an integrated web application server [4] that is running a web services engine.

The integrated web application server addresses the need for a minimal footprint, easy to configure, secure infrastructure for hosting web applications and web services. The current version of the server is based on the IBM WebSphere Liberty profile, a highly composable, dynamic application server.

Figure 12 on page 50 illustrates the underlying architecture of the integrated web services server.

    - Cf https://openliberty.io/

Liberty ?

- Profitons des possibilités de Liberty
  - Pour générer des jetons JWT
  - Pour valider des jetons JWT (via TAI)

- Possible également via le code d'implémentation RPG/COBOL

JWT ?

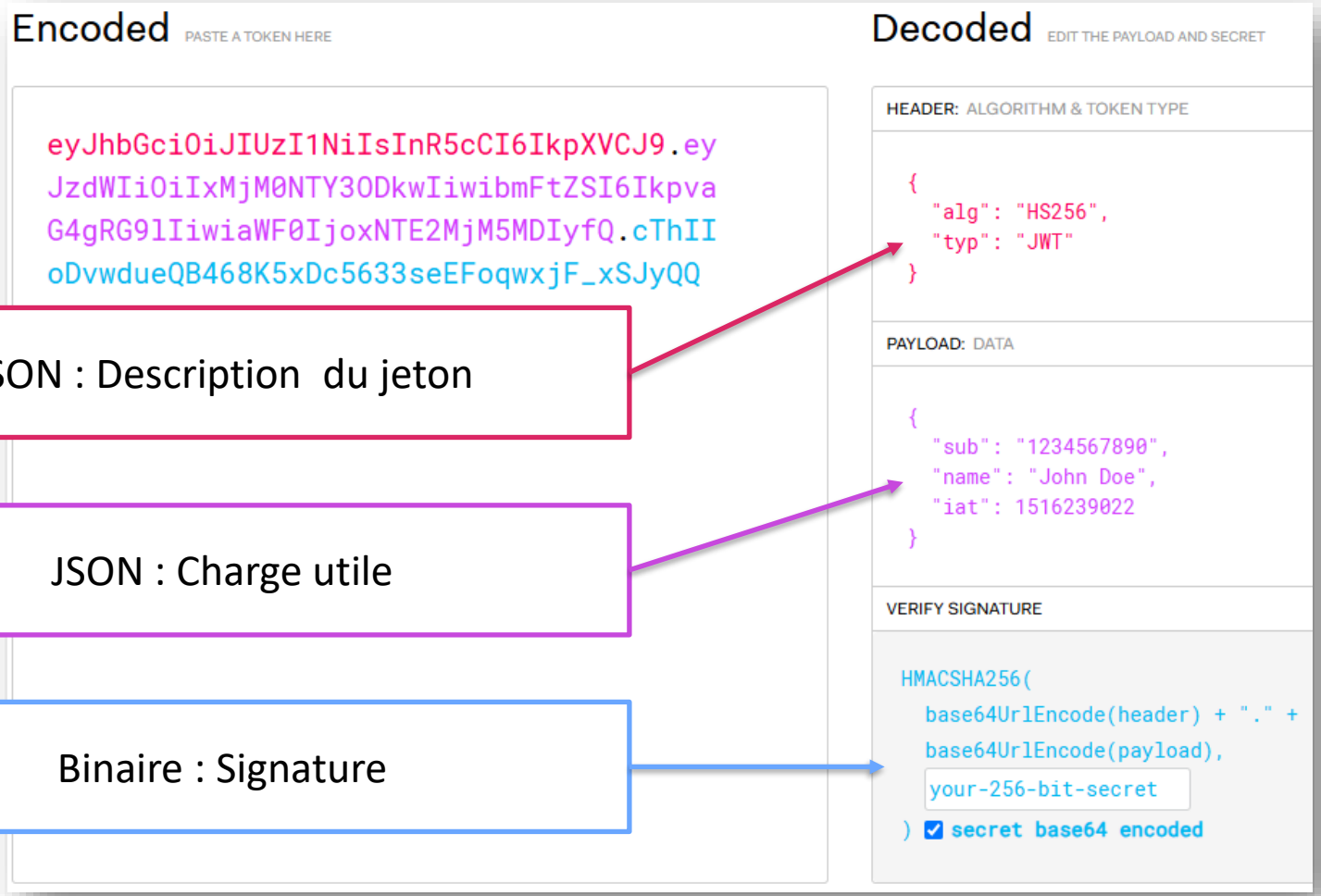
- JSON Web Token
  - https://fr.wikipedia.org/wiki/JSON_Web_Token

> **JSON Web Token** (JWT) est un standard ouvert défini dans la RFC 7519[1]. Il permet l'échange sécurisé de jetons (tokens) entre plusieurs parties. Cette sécurité de l'échange se traduit par la vérification de l'intégrité et de *l'authenticité* des données. Elle s'effectue par l'algorithme HMAC ou RSA.

  - Basé sur des protocoles simples et standards
    - JSON, BASE64, HMAC/RSA

JWT ?

- Structure



JWT ?

- Structure

JWT ?

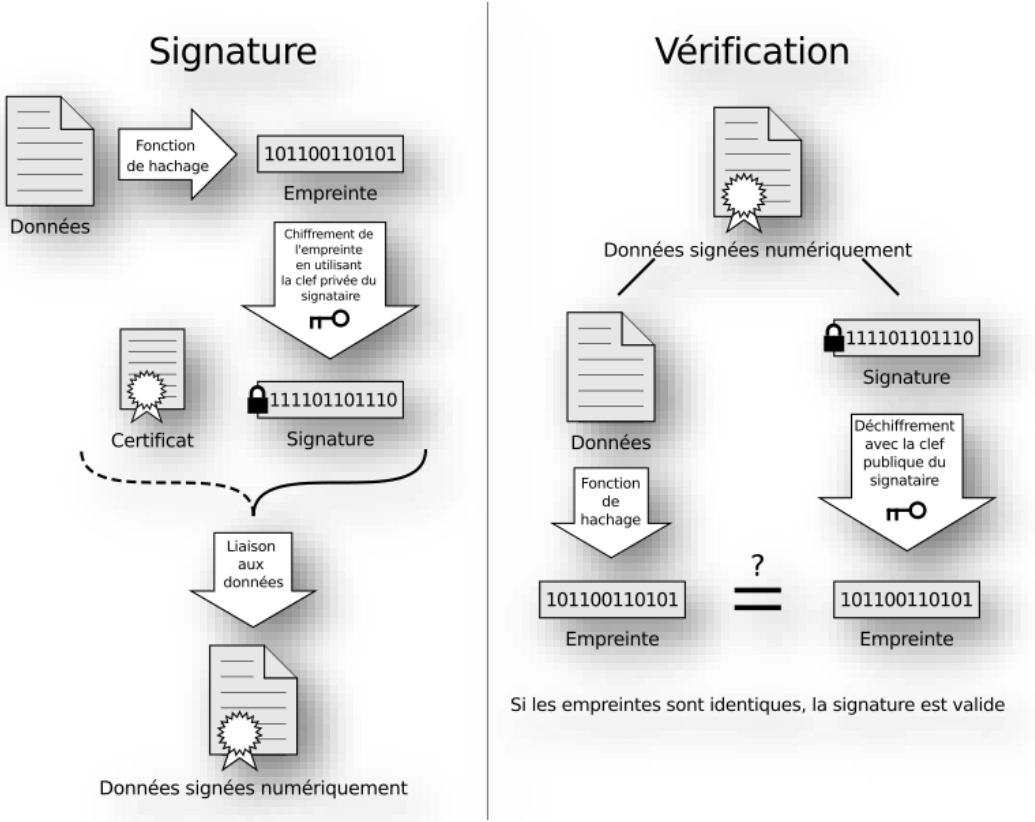
- Le payload contient des données standardisées, toutes facultatives, des claims :
  - iss : créateur (issuer) du jeton
  - sub : sujet (subject) du jeton
  - aud : audience du jeton
  - exp : date d'expiration du jeton
  - nbf : date avant laquelle le jeton ne doit pas être considéré comme valide (not before)
  - iat : date à laquelle a été créé le jeton (issued at)
  - jti : identifiant unique du jeton (JWT ID)
- Vous pouvez ajouter vos données personnalisées
  - Scope du jeton
  - Authentification interne
  - …

JWT ?

- Propriétés
  - Stateless
  - Authentifier un utilisateur pour un usage et pour une durée
  - La signature, via échange de clé préalable, permet de valider le jeton : authenticité et intégrité
  - Sans serveur tiers
  - Peut être encrypté
  - Supporte SSO, access control API …

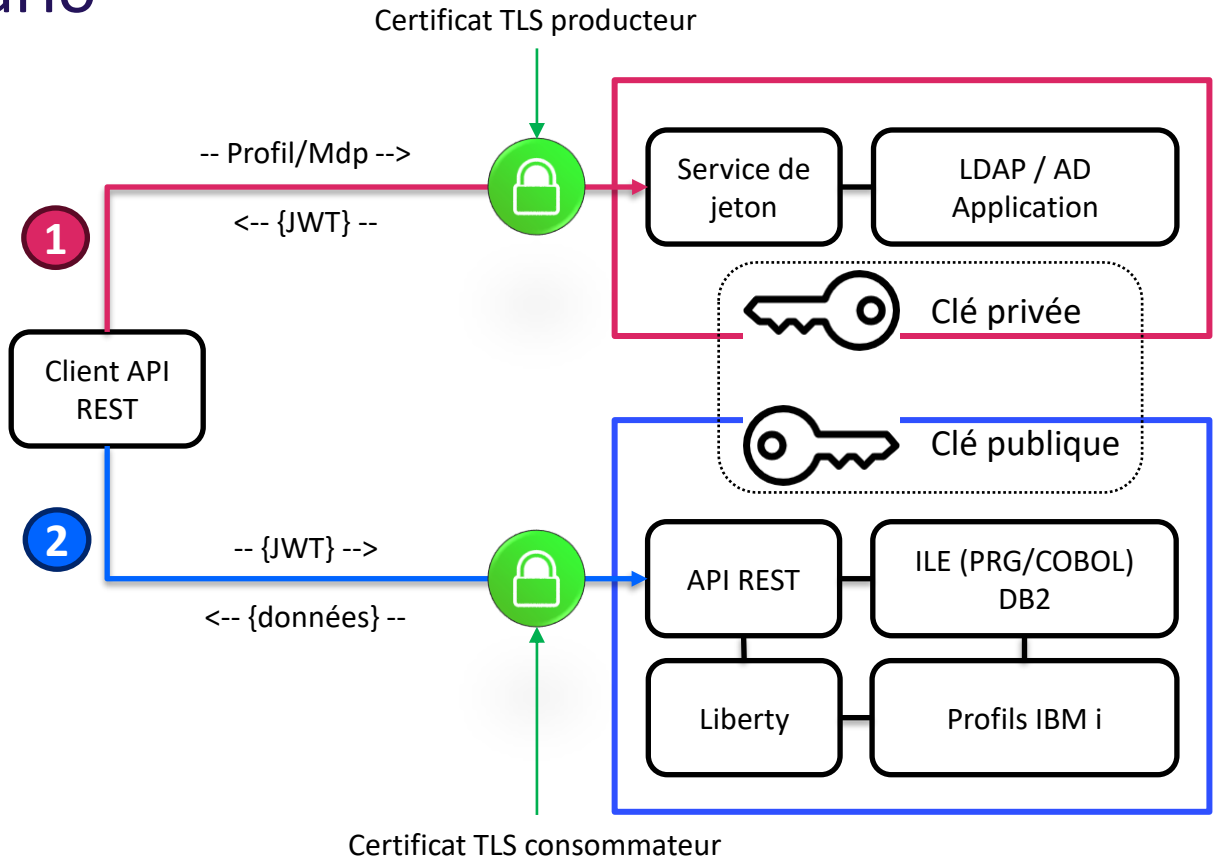
JWT

- Signature par clé asymétrique
  - Clé privée pour signer le jeton
  - Clé publique pour le vérifier



https://commons.wikimedia.org/wiki/File:Digital_Signature_diagram_fr.svg

Scénario


Certificat TLS producteur
-- Profil/Mdp -->
<-- {JWT} --
1
Client API REST
2
-- {JWT} -->
<-- {données} --
Service de jeton
LDAP / AD Application
Clé privée
Clé publique
API REST
ILE (PRG/COBOL) DB2
Liberty
Profils IBM i
Certificat TLS consommateur

IBM i
continuous innovation
continuous integration

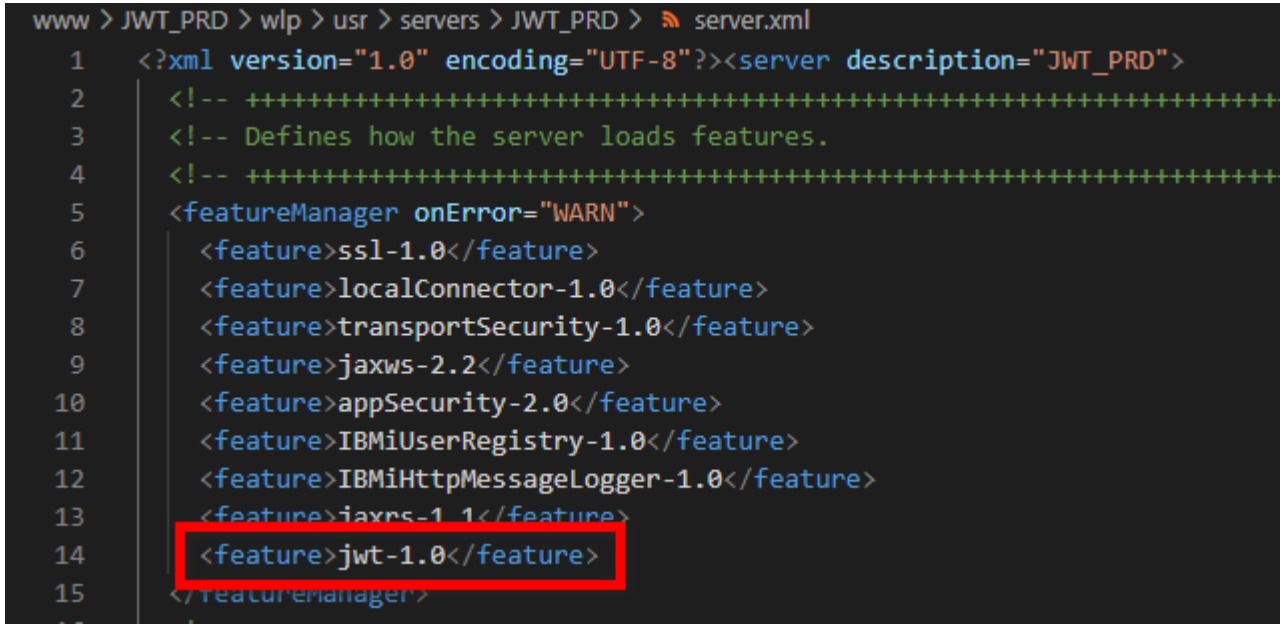
IWS serveur de jetons

Prérequis

- Nécessite TLS
- Sur l’instance Java
- Serveur HTTP Apache non nécessaire ici

Activation de la fonction jwt

- Fichier \www\{JWT_PRD}\wlp\usr\servers\{JWT_PRD}\server.xml
  - Ajouter la ligne

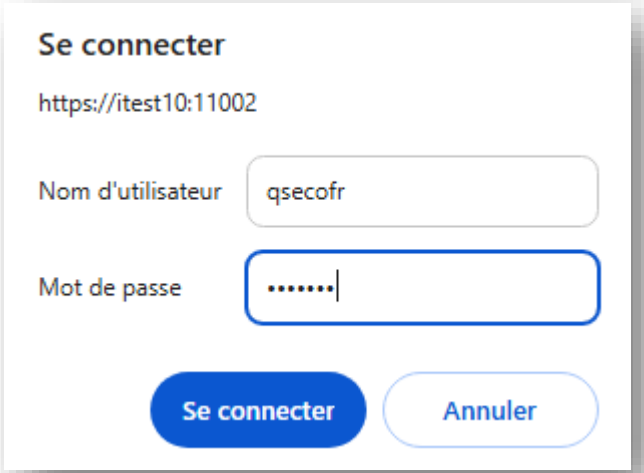


Activation de la fonction jwt

- Cela active une nouvelle URL disponible dans l'instance Java
  - https://itest10:11002/jwt/*

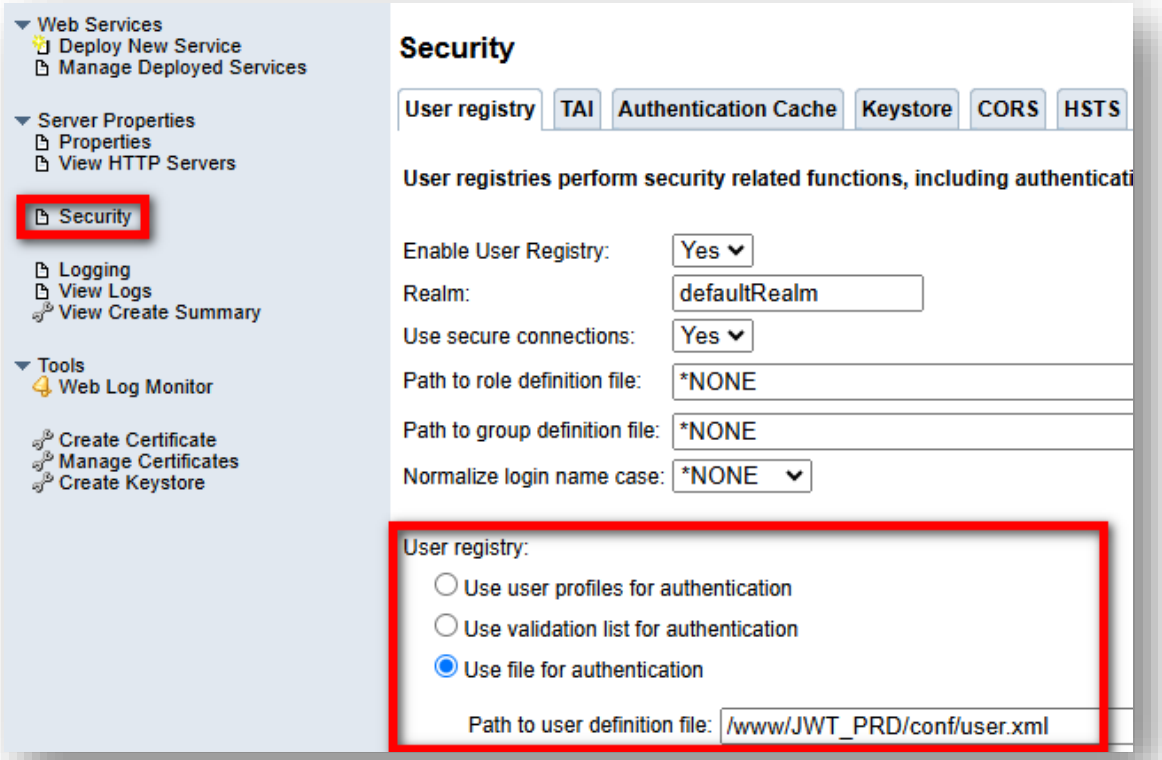
- Mais aussi une URL pour génération de JWT
  - https://<host>:<port>/jwt/ibm/api/<configId>/token
  - https://itest10:11002/jwt/ibm/api/uii24/token

  - Demande une authentification basique
    - Raison pour laquelle TLS est obligatoire



Activation de la fonction jwt

- Il faut configurer l'instance Java pour l'authentification
  - User Profiles
    - Déconseillé
  - Liste de validation
    - **Ne fonctionne pas ici ?**
  - Fichiers XML dans l'IFS
    - Notre solution



Activation de la fonction jwt

- Fichiers des utilisateurs autorisés
  - Ici /www/JWT_PRD/conf/users.xml

www > JWT_PRD > conf > user.xml

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <server>
3      <IBMiUserRegistry>
4          <user name="jwt"          password="abc123"/>
5          <user name="jwtuserall"   password="jwtuserall"/>
6          <user name="nb"           password="pwdnb"/>
7          <user name="jms"          password="{xor}Lz4sLChvLTs="/>
8      </IBMiUserRegistry>
9  </server>
```

  - Attention aux droits sur le fichier

Utilisat	Droits sur données
*PUBLIC	*EXCLUDE
QTMHHTTP	*RX
QWSERVICE	*RX

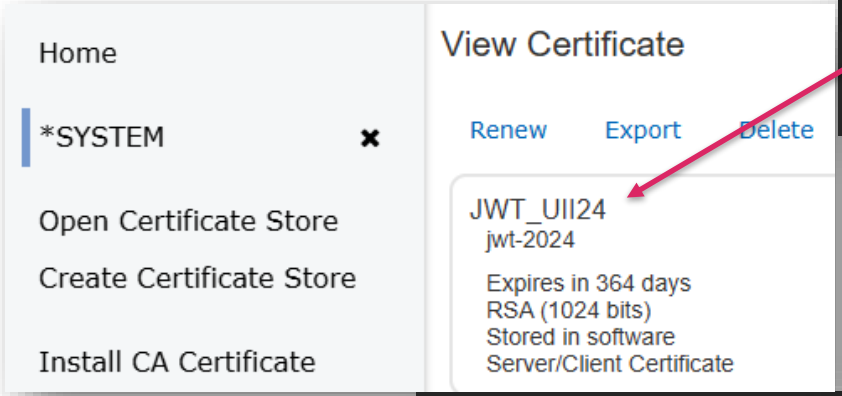
Personnalisation des JWT

- Toujours dans \www\{JWT_PRD}\wlp\usr\servers\{JWT_PRD}\server.xml
  - id
    - Identifie le constructeur de JWT
      - On peut en faire plusieurs avec des configurations diverses
    - Fait partie de l'URL : https://itest10:11002/jwt/ibm/api/uii24/token

```
<jwtBuilder id="uii24"
            keyAlias="JWT_UII24" keyStoreRef="defaultKeyStore"
            expiresInSeconds="3600s"
            audiences="uii2024,nb"
            jti="true"
            issuer="https://jwt.gaia.lan"/>
```

Personnalisation des JWT

- keyAlias & keyStoreRef
  - certificat utilisé pour calcul de la signature (doit supporter RSA)
    - keyAlias = nom du certificat (peut être différent de celui associé au serveur)




```
<jwtBuilder id="uii24"
            keyAlias="JWT_UII24" keyStoreRef="defaultKeyStore"
            expiresInSeconds="3600s"
            audiences="uii2024,nb"
            jti="true"
            issuer="https://jwt.gaia.lan"/>
```

```
<keyStore id="defaultKeyStore"
          location="/QIBM/UserData/ICSS/Cert/Server/DEFAULT.KDB" password="{xor}Njo2bmZmbQ=="
          provider="IBMi5OSJSSEProvider" type="IBMi5OSKeyStore"/>
```

Personnalisation des JWT

- expiresInSeconds, audiences, jti, issuer …
  - Propriétés à insérer dans le jeton
- Liste ici : https://www.ibm.com/docs/en/was-liberty/nd?topic=configuration-jwtbuilder

```
<jwtBuilder id="uii24"
            keyAlias="JWT_UII24" keyStoreRef="defaultKeyStore"
            expiresInSeconds="3600s"
            audiences="uii2024,nb"
            jti="true"
            issuer="https://jwt.gaia.lan"/>
```

JWT Builder (jwtBuilder)

Last Updated: 2023-11-16

Information about configuring the builder. The elements and attributes that you specify are used to build the token.

Name	Type	Default	Description
audiences	string		The trusted audience list to be included in the aud claim in the JSON web token.
claims	string		Specify a comma separated list of claims to include in the token. These claims must be existing user attributes that are defined for the subject of the JWT in the user registry.
contentEncryptionAlgorithm	– A256GCM		Specifies the encryption algorithm that is used to encrypt the JWT plaintext to produce the JWE ciphertext. **A256GCM** Use the AES GCM algorithm with a 256-bit key to encrypt the JWT plaintext of a JWE.
expiresInSeconds	A period of time with second precision	-1	Indicates the token expiration time in seconds. Takes precedence over expiry. When this attribute is set to a negative number, the value of the expiry attribute is used. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). For example, specify 30 seconds as 30s. You can include multiple values in a single entry. For example, 1m30s is equivalent to 90 seconds.
expiry	A period of time with hour precision	2h	Indicates the token expiration time in hours. ExpiresInSeconds takes precedence if present. Specify a positive integer followed by the unit of time, which can be hours (h). For example, specify 12 hours as 12h.
id	string	defaultJWT	This ID is used to identify the JWT builder. If an ID value is not specified, the builder is not processed. The ID must be a URL-safe string. The ID is used as part of the issuer value if the issuer configuration attribute is not specified. The JwtBuilder API uses this ID to determine which builder configuration to use to construct JWTs.
			An Issuer is a case-sensitive URL using the HTTP or HTTPS scheme that contains scheme, host

Obtenir un jeton

- https://itest10:11002/jwt/ibm/api/uii24/token
- Produit


```
{"token":
"eyJraWQiOiJDRm9kVG5rSTNhUHM2UkYwR0owcU5ZUlRGYTdSM1NST3M1MlZERTdldkswIiwidHlwI
joiSldUIiwiYWxnIjoiUlMyNTYifQ.eyJ0b2tlbl90eXBlIjoiQmVhcmVyIiwiYXVkIjpbInV1aTIw
MjQiLCJuYiJdLCJzdWIiOiJqd3QiLCJ1cG4iOiJqd3QiLCJyZWFsbSI6ImRlZmF1bHRSZWFsbSIsIm
p0aSI6InJPS2IwU2llYnZUM0tIc3EiLCJpc3MiOiJodHRwczovL2p3dC5nYWlhLmxhbiIsImV4cCI6
MTczMTYyNjE2MywiaWF0IjoxNzMxNjI2MTAzfQ.PEiyTOdiDdY1d1Qds4mckiWvnRCwy6-
EGd1HEcaJYmQMmF5kUALqRdeUbkf2DGQZDmY_BNdf9TYHu_uKT878PbiV3aVK0448iEzI3xuNWqhQn
NDlbnGqmYJtyuuubOMOljq6PU_YmvB3wqEiY7ICT4JLq3R0SPy5mvPhC2fLh3MrxOtRiyRahUZqT4u
J9qpfV_0gDR6RePYozkb1LiYSKwqulKCCXgxtMXcuhMlLo8zcAdEMFkXz-OE3NN-
IobIHbqti6pLaalKIh-
F0XFayPgL_ER0wtDoGI3Bltswu0UETm4VLF10Jd450iyuHh83pYCGAsOZRKpaGobwDb4U8jA"}
```

Obtenir un jeton

- Analysons avec https://jwt.io/

Encoded PASTE A TOKEN HERE


```
eyJraWQiOiJDRm9kVG5rSTNhUHM2UkYwR0owcU5
ZUlRGYTdSM1NST3M1MlZERTdldkswIiwidHlwIj
oiSldUIiwiYWxnIjoiUlMyNTYifQ.eyJ0b2tlbl
90eXBlIjoiQmVhcmVyIiwiYXVkIjpbInV1aTIwM
jQiLCJuYiJdLCJzdWIiOiJqd3QiLCJ1cG4iOiJq
d3QiLCJyZWFsbSI6ImRlZmF1bHRSZWFsbSIsImp
0aSI6InJPS2IwU2llYnZUM0tIc3EiLCJpc3MiOi
JodHRwczovL2p3dC5nYWlhLmxhbiIsImV4cCI6M
TczMTYyNjE2MywiaWF0IjoxNzMxNjI2MTAzfQ.P
EiyTOdiDdY1d1Qds4mckiWvnRCwy6-
EGd1HEcaJYmQMmF5kUALqRdeUbkf2DGQZDmY_BN
df9TYHu_uKT878PbiV3aVK0448iEzI3xuNWqhQn
NDlbnGqmYJtyuuubOMOljq6PU_YmvB3wqEiY7IC
T4JLq3R0SPy5mvPhC2fLh3MrxOtRiyRahUZqT4u
J9qpfV_0gDR6RePYozkb1LiYSKwqulKCCXgxtMX
cuhMlLo8zcAdEMFkXz-OE3NN-
IobIHbqti6pLaalKIh-
F0XFayPgL_ER0wtDoGI3Bltswu0UETm4VLF10Jd
450iyuHh83pYCGAsOZRKpaGobwDb4U8jA
```


HEADER: ALGORITHM & TOKEN TYPE


```
{
  "kid": "CFodTnkI3aPs6RF0GJ0qNYRTFa7R3SROs52VDE7evK0",
  "typ": "JWT",
  "alg": "RS256"
}
```


kid : (optionnel dans la norme) key id
typ : correspond à application/jwt de l'IANA
alg : pour signature

Obtenir un jeton

- Analysons avec https://jwt.io/

PAYLOAD: DATA


```
{
  "token_type": "Bearer",
  "aud": [
    "uui2024",
    "nb"
  ],
  "sub": "jwt",
  "upn": "jwt",
  "realm": "defaultRealm",
  "jti": "rOKb0SiebvT3KHsq",
  "iss": "https://jwt.gaia.lan",
  "exp": 1731626163,
  "iat": 1731626103
}
```


```
<jwtBuilder id="uii24"
            keyAlias="JWT_UII24" keyStoreRef="defaultKeyStore"
            expiresInSeconds="3600s"
            audiences="uii2024,nb"
            jti="true"
            issuer="https://jwt.gaia.lan"/>
```

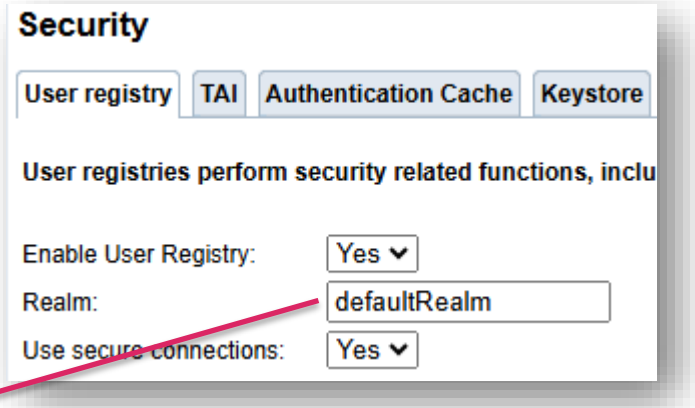
aud : audiences dans fichier de conf
sub & upn : utilisateur (jwt dans notre exemple)
jti : identifiant unique de jeton (pour jeton à usage unique par exemple)
iss : issuer
exp : expiration (secondes depuis EPOCH)
iat : date de création (secondes depuis EPOCH)

Obtenir un jeton

- Analysons avec https://jwt.io/

PAYLOAD: DATA


```
{
  "token_type": "Bearer",
  "aud": [
    "uui2024",
    "nb"
  ],
  "sub": "jwt",
  "upn": "jwt",
  "realm": "defaultRealm",
  "jti": "rOKb0SiebvT3KHsq",
  "iss": "https://jwt.gaia.lan",
  "exp": 1731626163,
  "iat": 1731626103
}
```

domaine

Obtenir un jeton

- Par SQL

```
1 values qsys2.http_get('https://itest10:11002/jwt/ibm/api/uii24/token', '{"basicAuth":"jwt,abc123"}');
2
```

```
00001
{"token": "eyJraWQiOiJDRm9kVG5rSTNhUHM2UkYwR0owcU5ZU1RGYTdSM1NST3M1M1ZERTdldkswIiwidHlwIjoiSldUIiwiYWxn...
```

- Facile !

www > JWT_PRD > conf > user.xml

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <server>
3      <IBMiUserRegistry>
4          <user name="jwt"          password="abc123"/>
5          <user name="jwtuserall"   password="jwtuserall"/>
6          <user name="nb"           password="pwdnb"/>
7          <user name="jms"          password="{xor}Lz4sLChvLTs="/>
8      </IBMiUserRegistry>
9  </server>
```

Dépannage

- Mauvais profil/mot de passe

  HTTP_STATUS_CODE = 401 (Unauthorized / Bad credentials)

  "WWW-Authenticate":"Basic realm=\"defaultRealm\""

- Sans TLS

  Error 404: CWWKS6052E: Un schéma HTTP est utilisé au point d'extrémité indiqué : http://itest10.gaia.lan:10002/s, HTTPS est requis.

- Invocation via le serveur Apache

  Not Found

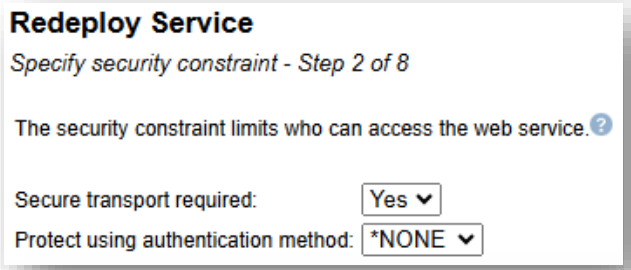
  The requested URL was not found on this server.

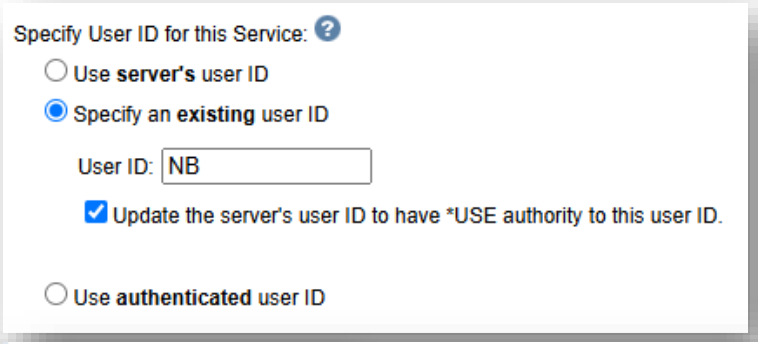
IBM i
continuous innovation
continuous integration

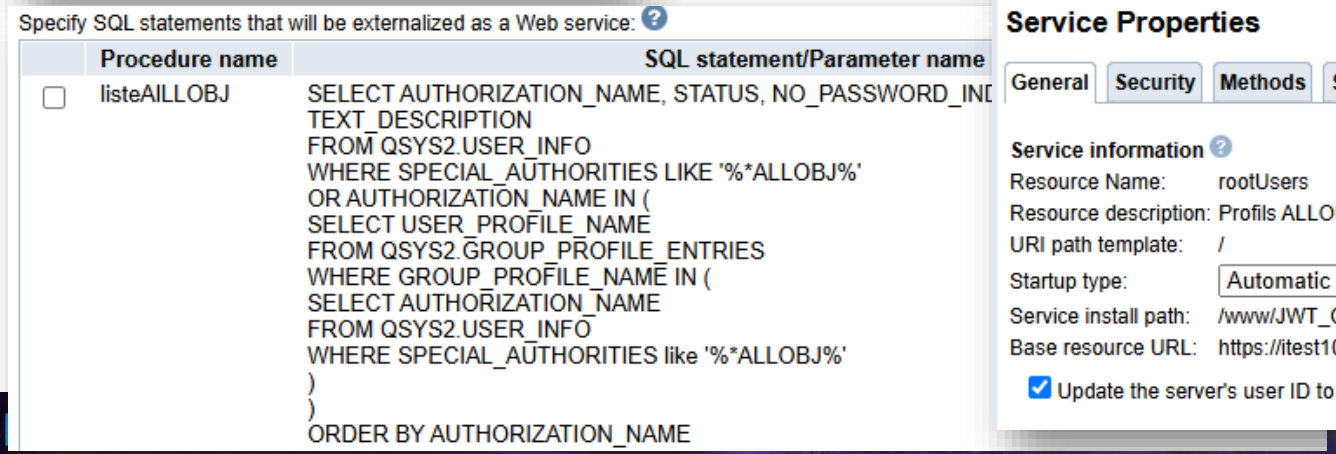
Authentification via JWT avec IWS

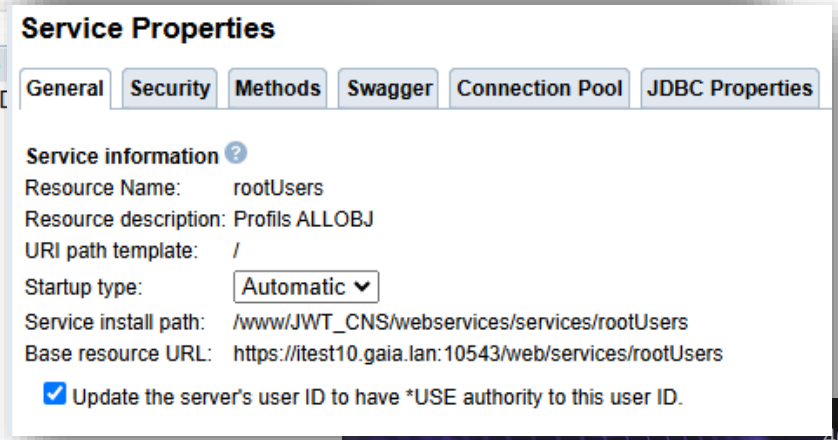
Notre service de test

- Déployer un service :
  - Liste des profils *ALLOBJ de la partition



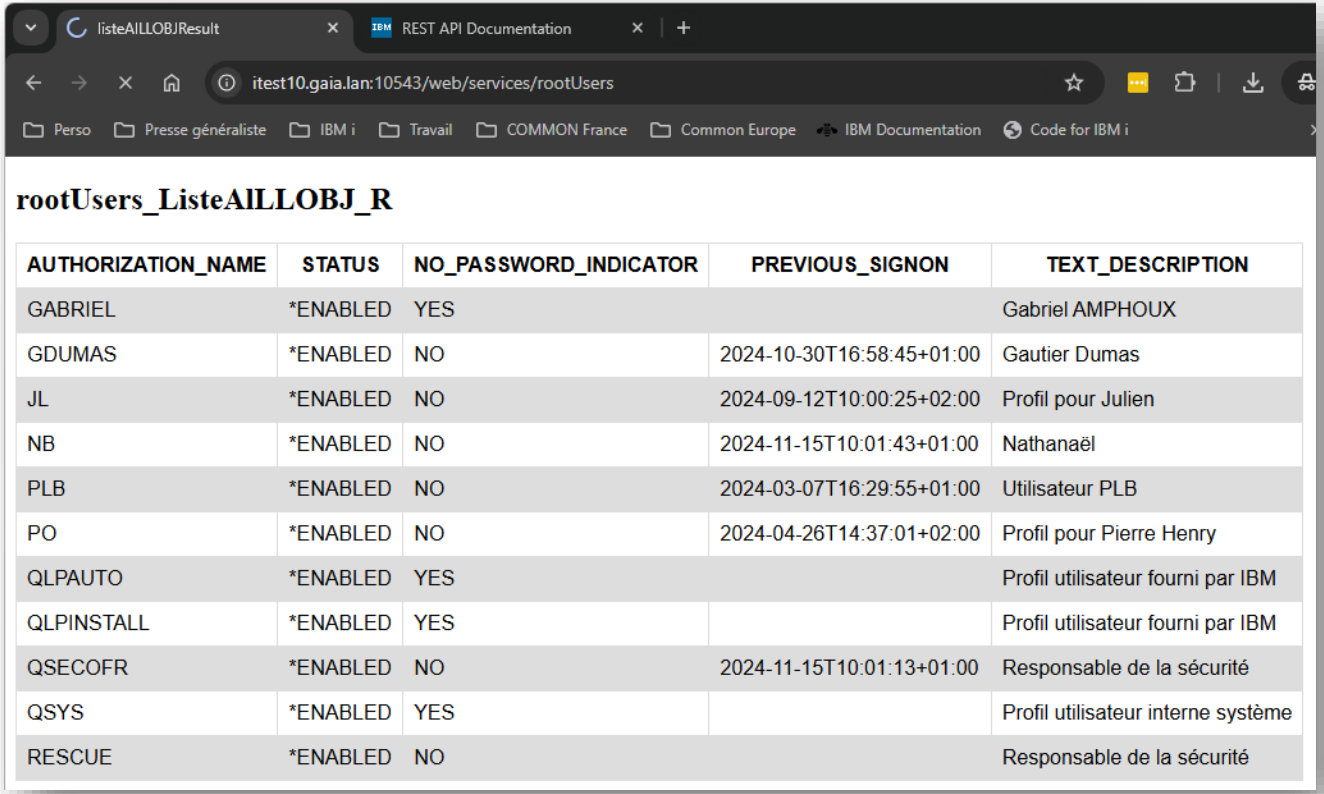






Notre service de test

- Produit
- A sécuriser !

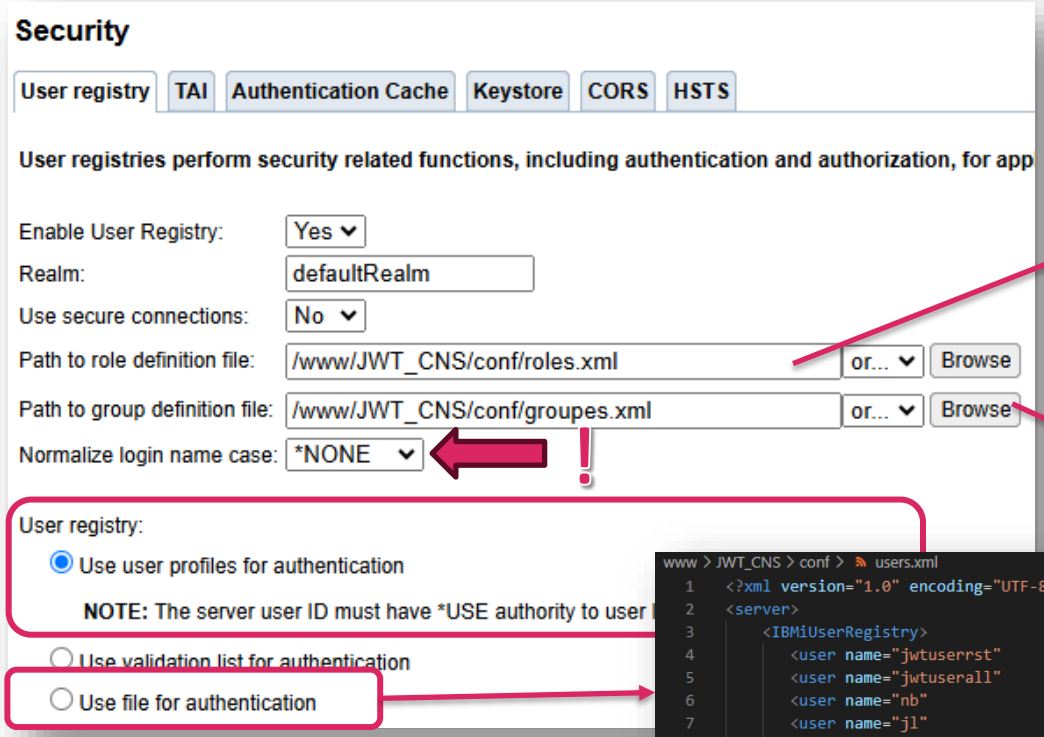


rootUsers_ListeAllLOBJ_R

AUTHORIZATION_NAME	STATUS	NO_PASSWORD_INDICATOR	PREVIOUS_SIGNON	TEXT_DESCRIPTION
GABRIEL	*ENABLED	YES		Gabriel AMPHOUX
GDUMAS	*ENABLED	NO	2024-10-30T16:58:45+01:00	Gautier Dumas
JL	*ENABLED	NO	2024-09-12T10:00:25+02:00	Profil pour Julien
NB	*ENABLED	NO	2024-11-15T10:01:43+01:00	Nathanaël
PLB	*ENABLED	NO	2024-03-07T16:29:55+01:00	Utilisateur PLB
PO	*ENABLED	NO	2024-04-26T14:37:01+02:00	Profil pour Pierre Henry
QLPAUTO	*ENABLED	YES		Profil utilisateur fourni par IBM
QLPINSTALL	*ENABLED	YES		Profil utilisateur fourni par IBM
QSECOFR	*ENABLED	NO	2024-11-15T10:01:13+01:00	Responsable de la sécurité
QSYS	*ENABLED	YES		Profil utilisateur interne système
RESCUE	*ENABLED	NO		Responsable de la sécurité

Etape 1 : authentification

- IWS permet la définition de rôles et de groupes pour l'authentification basique



www > JWT_CNS > conf > roles.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<server>
  <IBMiUserRegistry>
    <role name="admin">
      <user name="nb" />
    </role>
    <role name="restreint">
      <user name="jwtuserrst" />
      <group name="partenaire" />
    </role>
    <role name="all">
      <user name="jwtuserall" />
      <group name="gaia" />
    </role>
  </IBMiUserRegistry>
</server>
```

www > JWT_CNS > conf > users.xml

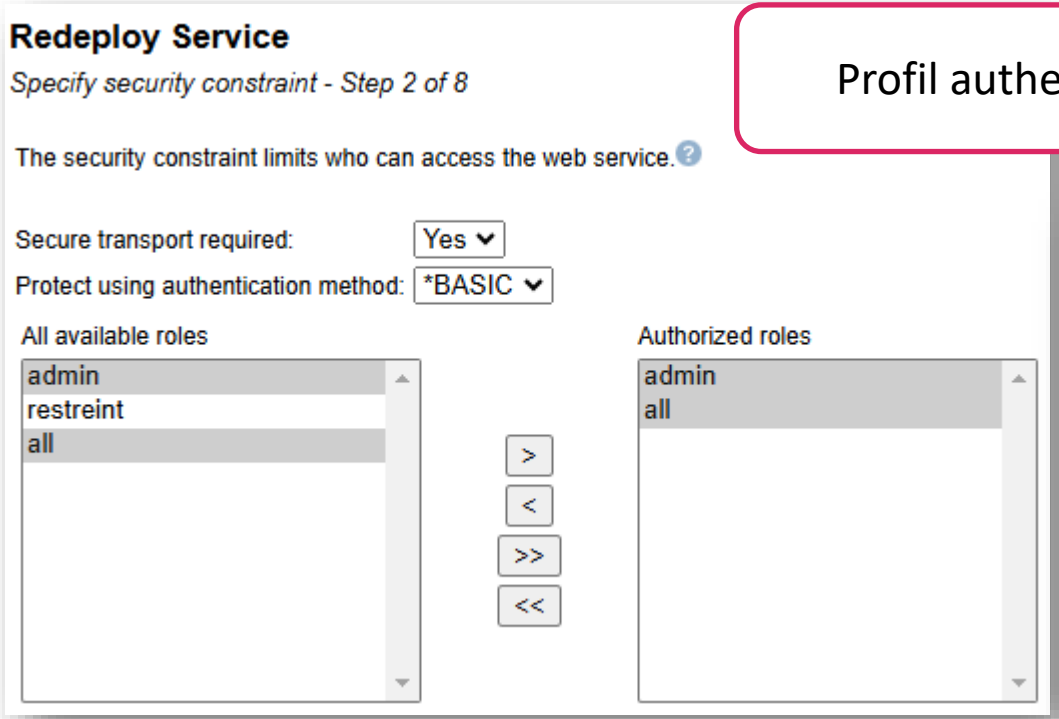
```xml
<?xml version="1.0" encoding="UTF-8"?>
<server>
    <IBMiUserRegistry>
        <user name="jwtuserrst"      password="jwtuserrst"/>
        <user name="jwtuserall"      password="jwtuserall"/>
        <user name="nb"              password="pwdnb"/>
        <user name="jl"              password="pwdjl"/>
        <user name="plb"             password="pwdplb"/>
        <user name="jms"             password="{xor}Lz4sLChvLTs="/>
    </IBMiUserRegistry>
</server>
```

www > JWT_CNS > conf > groupes.xml

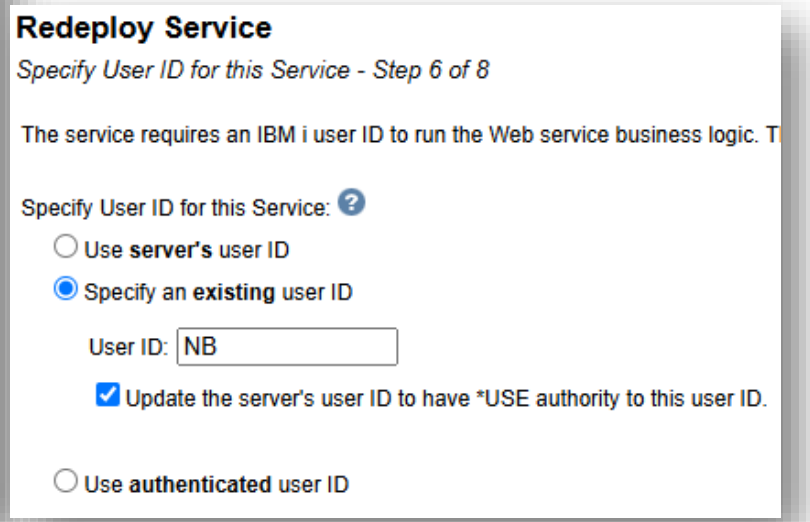
```xml
<?xml version="1.0" encoding="UTF-8"?>
<server>
  <IBMiUserRegistry>
    <group name="gaia">
      <member name="nb" />
      <member name="plb" />
      <member name="jl" />
      <member name="jwtuser" />
    </group>
    <group name="partenaire">
      <member name="gabriel" />
      <member name="gd" />
    </group>
  </IBMiUserRegistry>
</server>
```

Etape 1 : authentification

- Au niveau du service



Profil authentifié ⇔ Profil exécution



Etape 1 : authentification basique (profil/mot de passe)

- Appel du service
  - Utilisateurs
    - NB : OK
    - JWTUSERRST : ERREUR
    - JWTUSERALL : OK

Casse en fonction de Normalize

- Si vous avez choisi « Use authenticated user ID »
  - Le profils serveur IWS doit avoir *USE
  - Sur **chaque profil** possible !

```
Objet . . . . . :   JWTUSERALL
  Bibliothèque . . :     QSYS
Type d'objet . . . :   *USRPRF



                          Droits
Utilisat     Groupe       sur objet
*PUBLIC                   *EXCLUDE
NB                        *ALL
JWTUSERALL                USER DEF
QWSERVICE                 *USE
```

Etape 2 : authentification via JWT

- Fichier \www\{JWT_CNS}\wlp\usr\servers\{JWT_CNS}\server.xml

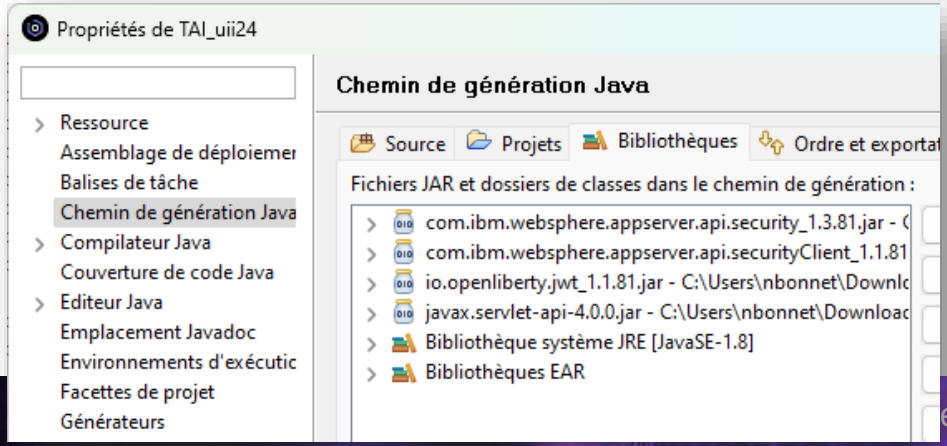
```
<featureManager onError="WARN">
  <feature>ssl-1.0</feature>
  <feature>localConnector-1.0</feature>
  <feature>transportSecurity-1.0</feature>
  <feature>jaxws-2.2</feature>
  <feature>appSecurity-2.0</feature>
  <feature>IBMiHttpMessageLogger-1.0</feature>
  <feature>jaxrs-1.1</feature>
  <feature>apiDiscovery-1.0</feature>
  <feature>IBMiUserRegistry-1.0</feature>
  <feature>jwt-1.0</feature>
</featureManager>
```

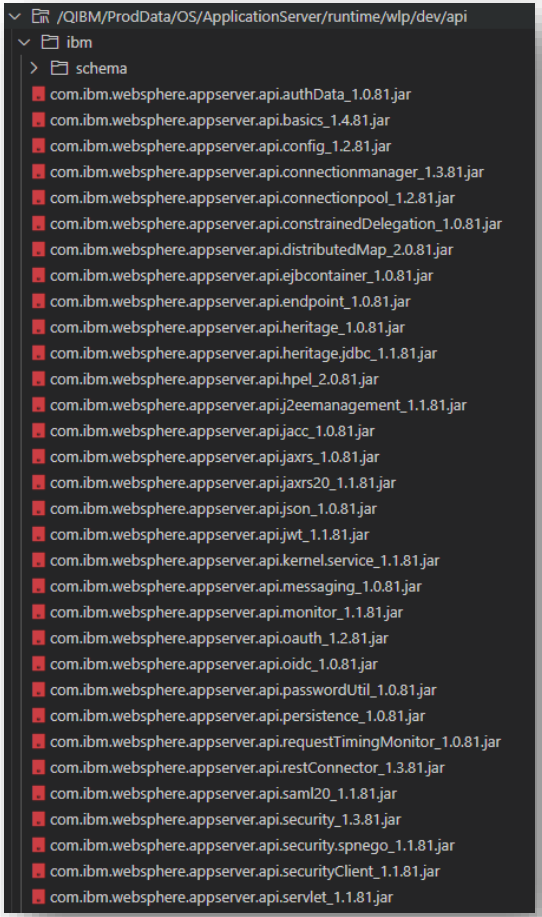
```
<jwtConsumer id="jwtcons"
             audiences="uii2024,nb"
             issuer="https://jwt.gaia.lan"
             trustStoreRef="defaultKeyStore"
             trustedAlias="JWT_UII24"/>
```

```
<keyStore id="defaultKeyStore"
          location="/QIBM/UserData/ICSS/Cert/Server/DEFAULT.KDB"
          password="{xor}Njo2bmZmbQ=="
          provider="IBMi5OSJSSEProvider"
          type="IBMi5OSKeyStore"/>
```

Etape 2 : authentification via JWT

- Classe Java pour TAI
  - Nécessite les dépendances :
    - com.ibm.websphere.appserver.api.security_1.3.xx.jar
    - com.ibm.websphere.appserver.api.securityClient_1.1.xx.jar
    - io.openliberty.jwt_1.xxx.jar
  - Présentes ici :
    - /QIBM/ProdData/OS/ApplicationServer/runtime/wlp/dev/api/ibm/





Etape 2 : authentification via JWT

- Code
  - Authentification = TLS !

```
public boolean isTargetInterceptor(HttpServletRequest req) throws WebTrustAssociationException {
    // Add logic to determine whether to intercept this request, i.e. this
    // TAI is going to authenticate the user

    boolean match = false;
    // Verify that the Authorization header contains a JWT
    if (req.getHeader("Authorization") != null) {
        System.out.println("[JWTTAI] HTTP header Authorization = " + req.getHeader("Authorization"));
        match = req.getHeader("Authorization")
                .matches("Bearer ([a-zA-Z0-9]|-|_)+\\.([a-zA-Z0-9]|-|_)+\\.([a-zA-Z0-9]|-|_|=)+");
        System.out.println("[JWTTAI] HTTP header Authorization match = " + match);
    }
    // If request is using HTTPS and does contain a JWT then intercept
    if (req.isSecure() && match) {
        return true;
    }
    return false;
}
```

Etape 2 : authentification via JWT

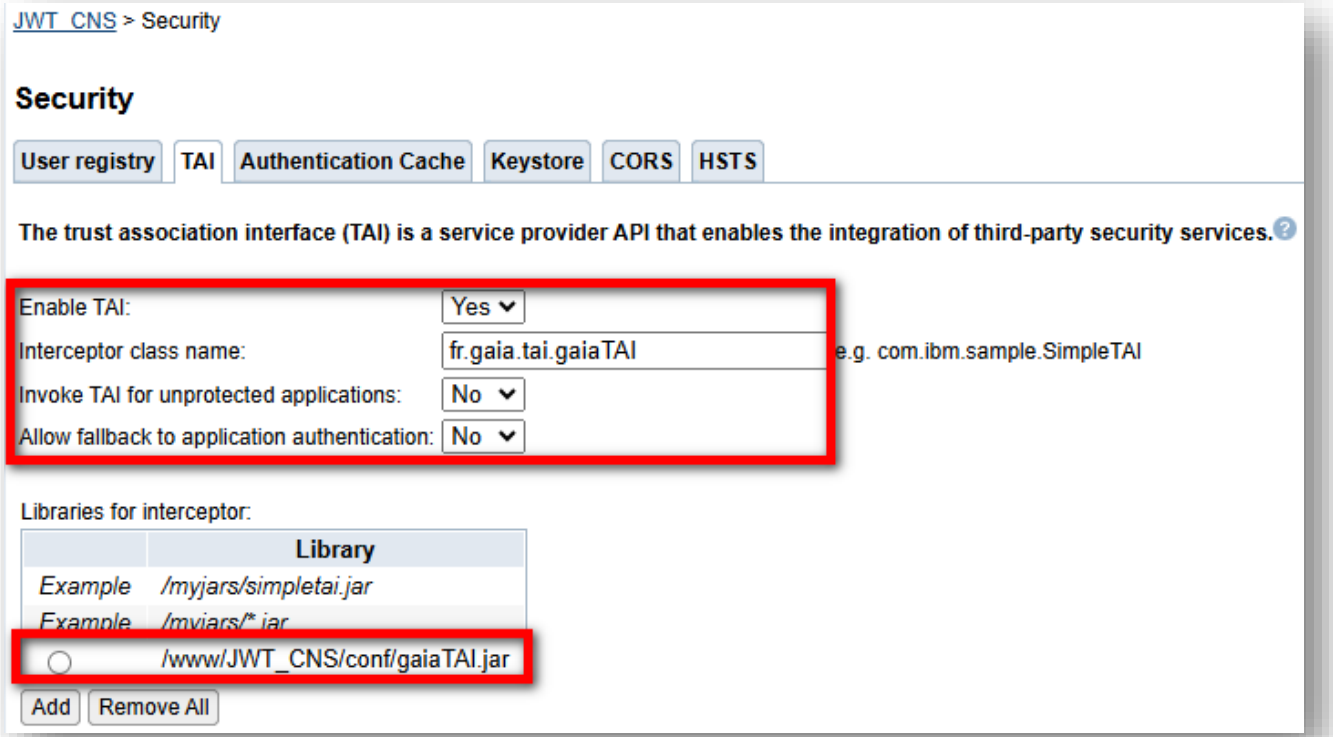
- Code

```
<jwtConsumer id="jwtcons"
             audiences="uii2024,nb"
             issuer="https://jwt.gaia.lan"
             trustStoreRef="defaultKeyStore"
             trustedAlias="JWT_UII24"/>
```

```
public TAIResult negotiateValidateandEstablishTrust(HttpServletRequest req, HttpServletResponse resp)
        throws WebTrustAssociationFailedException {
    // Add logic to authenticate an user and return a TAI result.

    // Get the Authorization header from the HTTP request and get the second
    // part (Bearer <token>)
    String tai_user = "";
    String jwtTokenString = req.getHeader("Authorization").split(" ")[1];

    // Decode the JWT using the JwtConsumer to obtain the claims
    JwtConsumer jwtConsumer;
    try {
        // Use the myJWTConsumer configuration (cf. server.xml)
        jwtConsumer = JwtConsumer.create("jwtcons");

        JwtToken jwtTokenConsumer = jwtConsumer.createJwt(jwtTokenString);
        Claims jwtClaims = jwtTokenConsumer.getClaims();
```

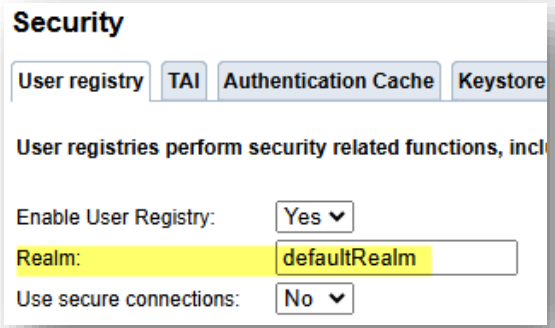
Etape 2 : authentification via JWT

- Exporter le .jar puis



Etape 2 : authentification via JWT

- Prérequis
  - Certificat
    - clé privée disponible pour le jwtBuilder, clé publique pour le jwtConsumer
  - Realm
    - Identique



  - Audiences
    - Au moins 1 audience en commun entre le jwtBuilder et jwtConsumer

```
<jwtBuilder id="uii24"
            keyAlias="JWT_UII24" keyStoreRef="defaultKeyStore"
            expiresInSeconds="3600s"
            audiences="uii2024,nb"
            jti="true"
            issuer="https://jwt.gaia.lan"/>
```

```
<jwtConsumer id="jwtcons"
             audiences="uii2024,nb"
             issuer="https://jwt.gaia.lan"
             trustStoreRef="defaultKeyStore"
             trustedAlias="JWT_UII24"/>
```

Avec du code

- Par SQL

```
-- ================================================================================
-- Obtenir un jeton :
-- ================================================================================
create or replace variable qgpl.jeton varchar(6000) ccsid 1208 ;
set qgpl.jeton  = json_value( qsys2.http_get('https://itest10:11002/jwt/ibm/api/uii24/token',
                                             '{"basicAuth":"jwtuserall,jwtuserall"}'),
                              '$.token' RETURNING varchar(6000) ) ;

-- Contrôles
values qgpl.jeton ;
```

```
00001
eyJraWQiOiJwbmxnQ2RRazg5TkRFWjNXcUlzdThqeXpSN0hScURuSWlwdldNTnlSc0ZrIiwidHlwIjoiSldUIiwiYWxnIjoiUlMyNTYifQ.eyJ0b2tlbl90eXBlIjoiQmVhcmVyIiwiYXVkIjoidWlpMjAyNCIsInN1YiI
```

```
-- ================================================================================
-- Utilisation jeton
-- ================================================================================

select * from table(qsys2.http_get_verbose('https://itest10.gaia.lan:10543/web/services/rootUsers')) ; -- => 401 non authentifié
select * from table(qsys2.http_get_verbose('https://itest10.gaia.lan:10543/web/services/rootUsers',
                                           '{"headers":{"accept":"application/json", "Authorization":"Bearer ' || trim(qgpl.jeton) || '"}}')) ;
```

RESPONSE_MESSAGE	RESPONSE_HTTP_HEADER
	{"HTTP_STATUS_CODE":401,"WWW-Authenticate":"Basic realm=\"defaultRealm\"","Content-Language":"fr-FR","C...

RESPONSE_MESSAGE	RESPONSE_HTTP_HEADER
{"rootUsers_ListeAllLOBJ_R" : [{ "AUTHORIZATION_NAME" : "GABRIEL", "STATUS" : "*ENABLED", "NO_PASSWORD_...	{"HTTP_STATUS_CODE":200,"Content-Type":"application/json",

Logs

- Cf /www/{JWT_CNS}/wlp/usr/servers/{JWT_CNS}/logs/messages.log

```
00000025 ibm.ws.security.authentication.internal.jaas.JAASServiceImpl I CWWKS1123I: Le plug-in d'authentification de collectivité avec le nom de classe NullCollectiveAuthenticationPlugin a été activé.
00000025 s.security.authentication.tai.internal.InterceptorConfigImpl I CWWKS1200I: L'initialisation de l'association de confiance a chargé trustAssociation//com.ibm.ws.security.authentication.tai.interceptor(interceptors)[default-0
00000025 s.security.authentication.tai.internal.InterceptorConfigImpl I CWWKS1201I: La classe fr.gaia.tai.gaiaTAI de l'initialisation de l'association de confiance a été chargée correctement.
```

- Par le code Java
  - Avec System.out.println

```
000000ec SystemOut                                    O [JWTTAI] HTTP header Authorization = Bearer eyJraWQiOiJwbmxnQ2RRazg5TkRFWjNXcU1zdThqe
000000ec SystemOut                                    O [JWTTAI] HTTP header Authorization match = true
000000ec SystemOut                                    O [JWTTAI] TAI USER = jwtuserall
```

  - Avec System.err.println

```
O [JWTTAI] HTTP header Authorization = Bearer eyJraWQiOiJwbmxnQ2RRazg5TkRFWjNXcU1zdThqeXpSN0hScURuSWlwd1dNTnlSc0ZrIiwidHlwIjoiSldUIiwiYWxnIjoiUlMyNTYifQ.eyJ0b2tlbl90eXBlIjoiQmVhcmVyIiwiYXVkIjoidWlpMjAyNCIsInN1YiI6Imp3dHVzZXJhbGwiLCJ1cG4iOiJqd3R1c2VyYWxs
O [JWTTAI] HTTP header Authorization match = true
R [JWTTAI] CWWKS6031E: Le consommateur JWT (JSON Web Token) [jwtcons] ne peut pas traiter la chaîne du jeton. CWWKS6045E: Un jeton JWT (JSON Web Token) avec les mêmes revendications 'iss' https://jwt.gaia.lan] et 'jti' [x46h3t7DnlZ2BxA1a déjà été reçu,
```

SystemErr R [JWTTAI] CWWKS6031E: Le consommateur JWT (JSON Web Token) [jwtcons] ne peut pas traiter la chaîne du jeton. CWWKS6045E: Un jeton JWT (JSON Web Token) avec les mêmes revendications 'iss' https://jwt.gaia.lan] et 'jti' [x46h3t7DnlZ2BxA1a déjà été reçu, ce qui pourrait dénoter une attaque par réinsertion. Assurez-vous que l'émetteur du jeton fournisse un jeton avec une revendication 'jti' unique.

Et ensuite ?

- Définition de plusieurs producteurs / consommateurs
  - Certificats et propriétés différentes
  - Mais le realm reste lié au serveur

https://itest10:11002/jwt/ibm/api/uii24/token

```
<jwtBuilder id="uii24"
            keyAlias="JWT_UII24" keyStoreRef="defaultKeyStore"
            expiresInSeconds="3600s"
            audiences="uii2024,nb"
            jti="true"
            issuer="https://jwt.gaia.lan"/>
<jwtBuilder id="uii24volubis"
            keyAlias="QIBM_HTTP_SERVER_JWT_PRD" keyStoreRef="defaultKeyStore"
            expiresInSeconds="600s"
            audiences="volubis"
            issuer="https://auth.volubis.fr"/>
```

https://itest10:11002/jwt/ibm/api/uii24volubis/token

Et ensuite ?

- Possibilité de « fallback » sur l'authentification basique en cas d'échec de l'authentification JWT
- Personnalisation du code Java
  - Intégrer vos propres règles d'accès
- Transmission d'informations issues du jeton à l'implémentation RPG
  - Via entêtes ou variables d'environnement
- Usage d'autres entêtes
  - X-forwarded-for
  - Personnalisées

Conclusion

- Un moyen simple de sécuriser vos services
  - Création et validation de jetons
  - En utilisant les outils classiques de l'IBM i : DCM
  - Nécessaire pour s'intégrer dans les outils standards (API Gateway, API Manager …)

  - Ne demande que quelques lignes de code Java
  - S'appuie sur des fonctions fournies et supportées par IBM

  - Surcoût en performances négligeable (non mesurable sur nos machines)

Références

- Liberty
  - https://www.ibm.com/docs/en/was-liberty/base?topic=management-liberty-features
  - https://www.ibm.com/docs/en/was-liberty/base?topic=cjwtil-building-consuming-json-web-token-jwt-tokens-in-liberty
  - https://www.ibm.com/docs/en/was-liberty/base?topic=liberty-obtaining-json-web-tokens-in
  - https://www.ibm.com/docs/en/was-liberty/nd?topic=configuration-jwtbuilder
  - https://www.ibm.com/docs/en/was-liberty/base?topic=configuration-jwtconsumer
  - https://www.ibm.com/docs/en/was-liberty/base?topic=api-websphere-security-jwt-builder-token
  - https://www.ibm.com/support/knowledgecenter/SSEQTP_liberty/com.ibm.websphere.javadoc.liberty.doc/com.ibm.websphere.appserver.api.security_1.3-javadoc/com/ibm/wsspi/security/tai/TrustAssociationInterceptor.html
- Projets Open Source
  - https://github.com/cicsdev/cics-java-liberty-tai-jwt
  - https://github.com/bmarolleau/iws-tai-security
- JWT
  - https://tools.ietf.org/html/rfc7519
  - https://jwt.io/
  - https://book.hacktricks.xyz/fr/pentesting-web/hacking-jwt-json-web-tokens
- Également
  - https://www.linkedin.com/in/johannes-heirman/


MERCI