

Université **IBM i**

19 et 20 novembre 2024

IBM Innovation Studio Paris

S18 – Services Web avec IWS : sécurité avancée

19 novembre 14:45 - 15:45

Nathanaël Bonnet

GAIA / VOLUBIS

nathanael.bonnet@gaia.fr



uui2024

#ibmi

#uui2024

The classic IBM logo, consisting of the letters 'IBM' in a bold, sans-serif font with horizontal stripes.The logo for 'Common France', featuring the word 'common' in a stylized, outlined font above the word 'FRANCE' in a smaller, simple font.



Nathanaël BONNET

IBM i depuis 1999

Expert IBM i



FRANCE



Université IBM i

19 et 20 novembre 2024

IBM i
continuous innovation
continuous integration



Centre de services IBM i

- Mutualisé
- A distance



Modernisation

- Accompagnement
- Prototypage



Expertise technique

- Prestations de service (audit, consulting...)
- Transfert de connaissances (workshops)



FORMATIONS

SUR SITE
A DISTANCE

INTER
SUR MESURE



COURS EN LIGNE

FORMATS COURS

REPLAYS EN LIGNE



BASE DE CONNAISSANCE

IBM i

EN LIGNE



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>



Services Web avec IWS : sécurité avancée

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

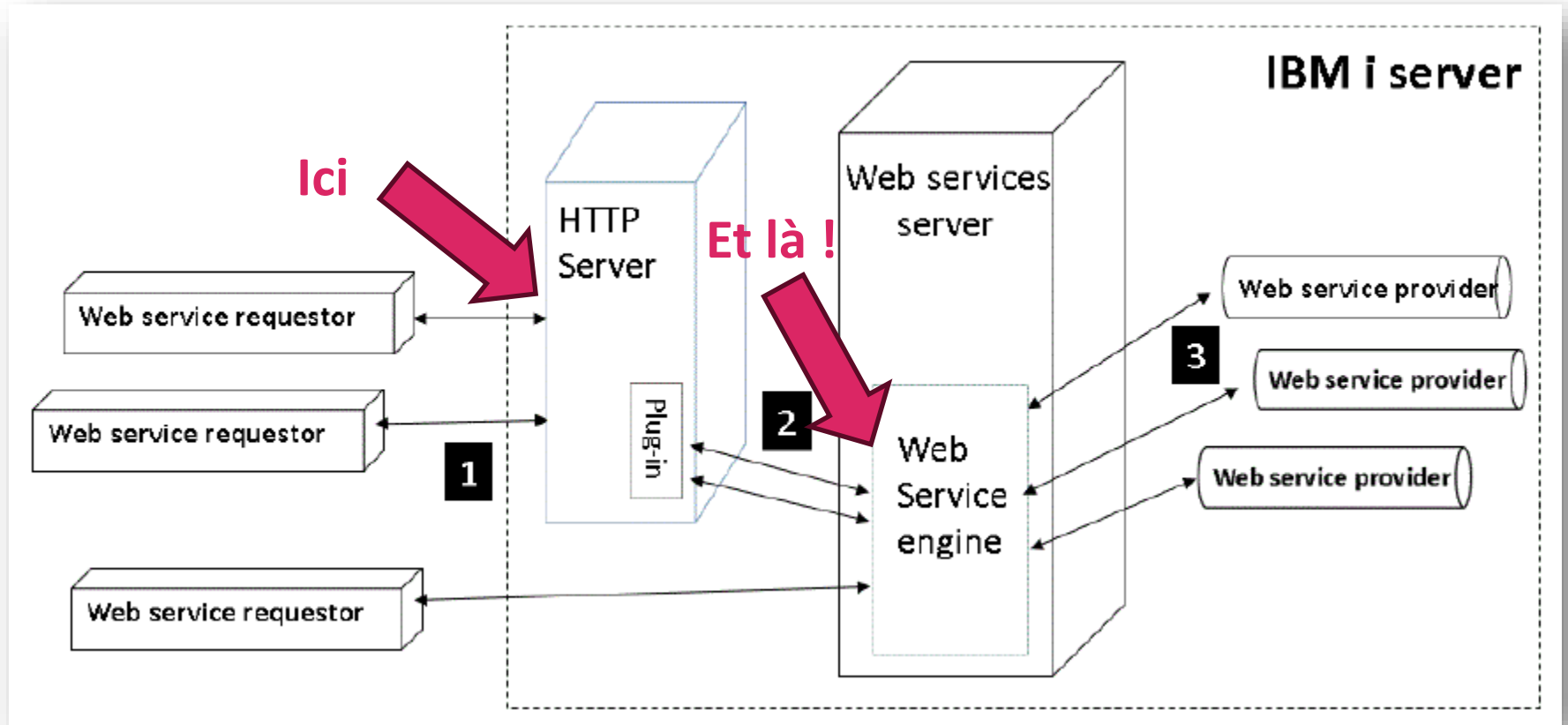
Université IBM i

19 et 20 novembre 2024



Bonnes pratiques

Certificat



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

The screenshot displays the IBM Web Administration console interface. On the left, a navigation pane shows the 'Configure TLS' option highlighted in red. The main content area shows the 'Configure TLS' wizard with the following steps:

- Specify Port Information - Step 2 of 7**

The port this HTTP server is currently listening on needs to be updated. Specify the port. This has no impact to the TLS port specified.

Specify TLS port for HTTP server: ?

TLS port:

Note: Most browsers make secure requests to port 443 by default.

Disable the non-TLS port?: ?

 - Yes, disable non-TLS port while configuring TLS port
 - No, leave non-TLS port enabled while still configuring TLS port
- Specify Digital Certificate - Step 4 of 7**

To configure TLS, your server must have a digital certificate. The wizard will create a new certificate for you.

Specify digital certificate for server: ?

 - Issue a new certificate by local CA
 - Select existing certificate from system certificate store
- Summary - Step 7 of 7**

When you click **Finish**, this wizard configures TLS for the HTTP server `JWT_CNS`. The HTTP server is associated with application server `QIBM_HTTP_SERVER_JWT_CNS`.

TLS port: 10443
Certificate Store: System (*SYSTEM) certificate store
Certificate Name: QIBM_HTTP_SERVER_JWT_CNS
Server Application Name: QIBM_HTTP_SERVER_JWT_CNS

You can use [Digital Certificate Manager](#) GUI interface to manage the server application and the associated certificate.

Note: When the configuration is completed, you need to restart the HTTP server and associated application server `JWT_CNS`.

Certificat – Serveur Java

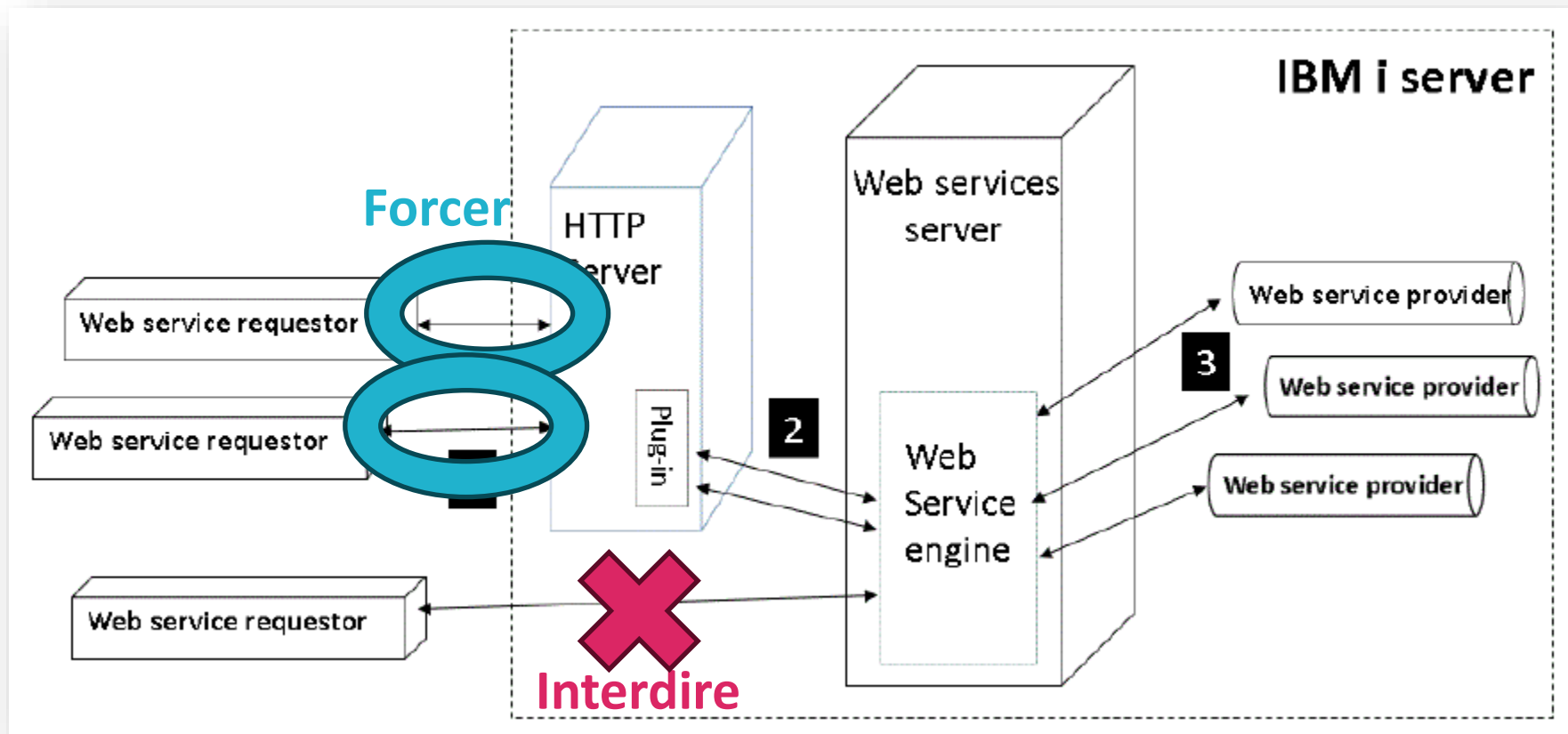
The screenshot displays the IBM Web Administration console interface for configuring TLS on a Java server. The main navigation bar includes 'Setup', 'Manage', 'Advanced', and 'Related Links'. The current server is identified as 'JWT_CNS - V2.6 (web services)'. The 'Manage' section is active, showing a tree view with 'Configure TLS' highlighted. Three overlapping wizard windows are visible:

- Configure TLS (Step 2 of 10):** Focuses on specifying the port and protocol. Fields include IP address (All IP addresses), TLS port (10543), and TLS protocol (TLSv1.2). A checkbox for 'Disable the non-TLS port?' is checked.
- Configure TLS (Step 3 of 10):** Focuses on specifying keystore information. The keystore path is '/www/jwt_cns/wlp/usr/servers/jwt_cns/resources/security/key.jks' and the keystore type is 'JKS'. The 'Specify keystore path and type' radio button is selected.
- Configure TLS (Step 6 of 10):** Focuses on specifying a digital certificate. The 'Select existing certificate from the keystore' radio button is selected. A dropdown menu shows available certificates, with 'QIBM_HTTP_SERVER_JWT_CNS' selected.

The final summary window shows the following configuration details:

IP address:	All IP addresses
TLS port:	10543
SSL protocol:	TLSv1.2
Keystore:	DCM SYSTEM store
Certificate Name:	QIBM_HTTP_SERVER_JWT_CNS

Endpoint



Endpoint

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

```
<httpEndpoint accessLoggingRef="defaultAccessLogging" host="*" httpOptionsRef="defaultHttpOptions" httpPort="10000" id="httpEndpoint0" />  
<httpEndpoint accessLoggingRef="defaultAccessLogging" host="127.0.0.1" httpPort="-1" httpsPort="10543" id="httpEndpoint1" />  
  <sslOptions sslRef="SSLSettingsByWebAdmin" />  
</httpEndpoint>
```

Application Server Ports JRE Web Services Subsystem Information

Port information for the integrated Web application server ?

Command port:

10014

	IP address or hostname	HTTP port	HTTPS port	TLS setting
Example	*	10000		
Example	10.1.2.3	10000	10001	mySSLSetting
<input type="radio"/>	*			
<input type="radio"/>	*		10543	SSLSettingsByWebAdmin

	IP address or hostname	HTTP port	HTTPS port	TLS setting
Example	*	10000		
Example	10.1.2.3	10000	10001	mySSLSetting
<input type="radio"/>	*			
<input checked="" type="radio"/>	127.0.0.1		10543	SSLSettingsByWebAdmin

Add

	Trusted HTTP proxy server IP addresses
Example	*
Example	10.1.2.3
Example	None
<input type="radio"/>	172.17.0.2
<input type="radio"/>	172.30.14.17
<input type="radio"/>	127.0.0.1

Add

Reset

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>



Ce site est inaccessible

itest10.gaia.lan n'autorise pas la connexion.

Voici quelques conseils :

- Vérifier la connexion
- Vérifier le proxy et le pare-feu

ERR_CONNECTION_REFUSED

Université IBM i

19 et 20 novembre 2024



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 7519¹. 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

Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MzkwMjQ.cThIIoDvwduQB468K5xDc5633seEFoqwxF_xSJyQQ
```

JSON : Description du jeton

JSON : Charge utile

Binaire : Signature

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{  
  "alg": "HS256",  
  "typ": "JWT"  
}
```

PAYLOAD: DATA

```
{  
  "sub": "1234567890",  
  "name": "John Doe",  
  "iat": 1516239022  
}
```

VERIFY SIGNATURE

```
HMACSHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  your-256-bit-secret  
)  secret base64 encoded
```

JWT ?

- Structure

Encoded

PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG91IiwiaWF0IjoxNTE2MzkwMjQ.cThIIoDvwdueQB468K5xDc5633seEFoqwxjF_xSJyQQ
```

↑

Encodage des parties du jeton en Base64url

Decoded

EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{  "alg": "HS256",  "typ": "JWT"}
```

PAYLOAD: DATA

```
{  "sub": "1234567890",  "name": "John Doe",  "iat": 1516239022}
```

VERIFY SIGNATURE

```
HMACSHA256(  base64UrlEncode(header) + "." +  base64UrlEncode(payload),  your-256-bit-secret)
```

secret base64 encoded

JWT ?

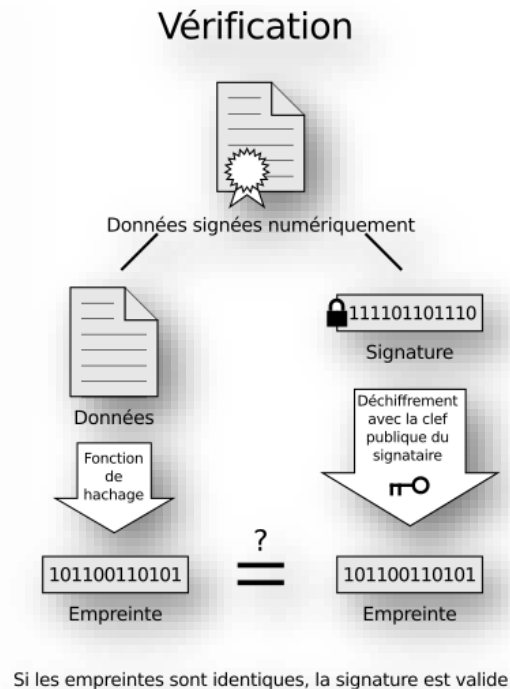
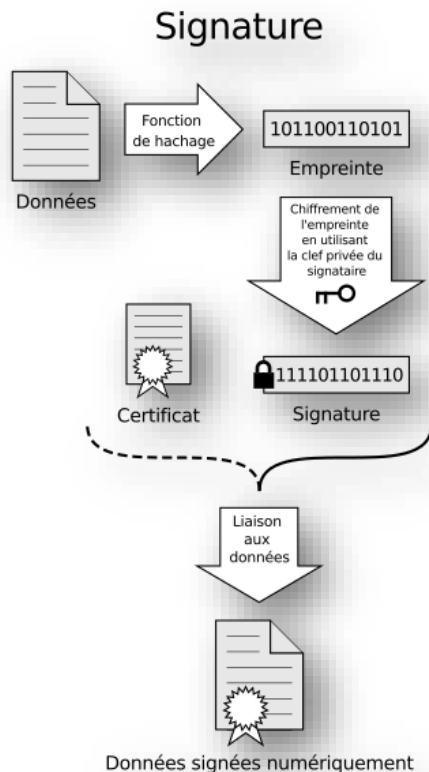
- Le payload contient des données standardisées, toutes facultatives, des claims :
 - iss : créateur (issuer) du jeton
 - sub : sujet (subject) du jeton
 - aud : audience du jeton
 - exp : date d'expiration du jeton
 - nbf : date avant laquelle le jeton ne doit pas être considéré comme valide (not before)
 - 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 ...

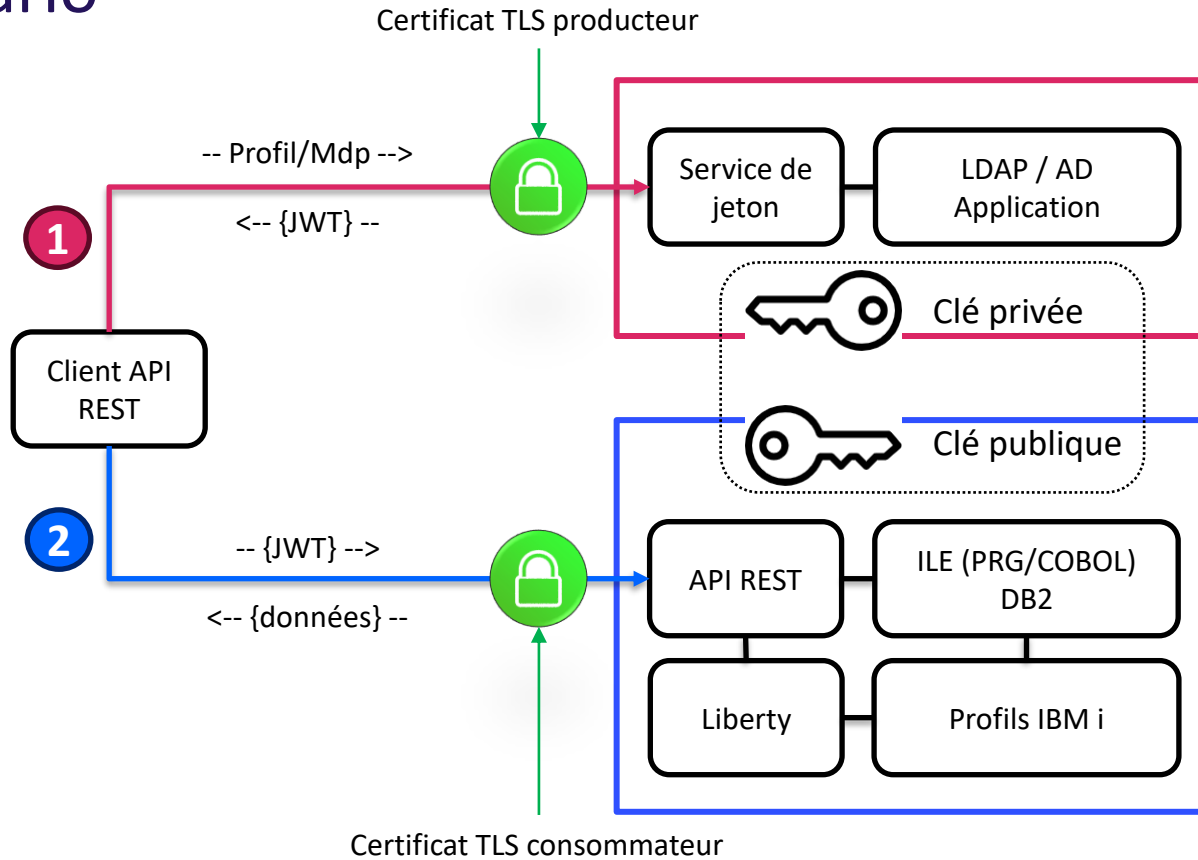
JWT

- 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





IWS serveur de jetons

Prérequis

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

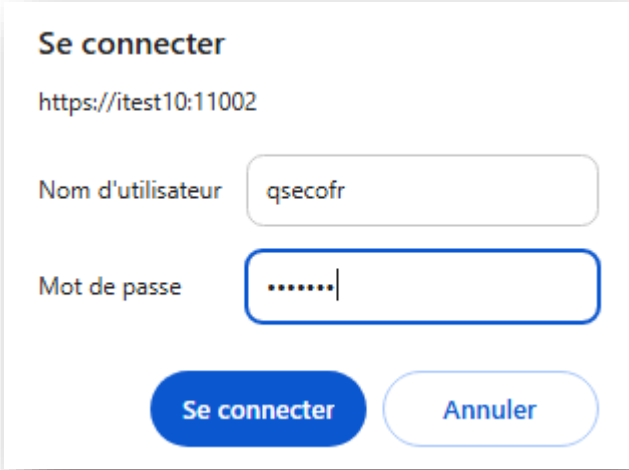
Activation de la fonction jwt

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

```
www > JWT_PRD > wlp > usr > servers > JWT_PRD > server.xml
1  <?xml version="1.0" encoding="UTF-8"?><server description="JWT_PRD">
2  <!-- ++++++
3  <!-- Defines how the server loads features.
4  <!-- ++++++
5  <featureManager onError="WARN">
6      <feature>ssl-1.0</feature>
7      <feature>localConnector-1.0</feature>
8      <feature>transportSecurity-1.0</feature>
9      <feature>jaxws-2.2</feature>
10     <feature>appSecurity-2.0</feature>
11     <feature>IBMiUserRegistry-1.0</feature>
12     <feature>IBMiHttpMessageLogger-1.0</feature>
13     <feature>jaxrs-1.1</feature>
14     <feature>jwt-1.0</feature>
15 </featureManager>
```


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



The image shows a login dialog box with the following elements:

- Se connecter** (Title)
- `https://itest10:11002` (URL)
- Nom d'utilisateur** (Label) with an input field containing `qsecofr`
- Mot de passe** (Label) with an input field containing seven dots and a cursor
- Se connecter** (Blue button)
- Annuler** (White button with blue border)

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

Security

User registries perform security related functions, including authentication

Enable User Registry: Yes ▾

Realm: defaultRealm

Use secure connections: Yes ▾

Path to role definition file: *NONE

Path to group definition file: *NONE

Normalize login name case: *NONE ▾

User registry:

- Use user profiles for authentication
- Use validation list for authentication
- Use file for authentication

Path to user definition file: /www/JWT_PRD/conf/user.xml

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
<u>*PUBLIC</u>	<u>*EXCLUDE</u>
QTMHHTTP	*RX
QWSERVICE	*RX

Personnalisation des JWT

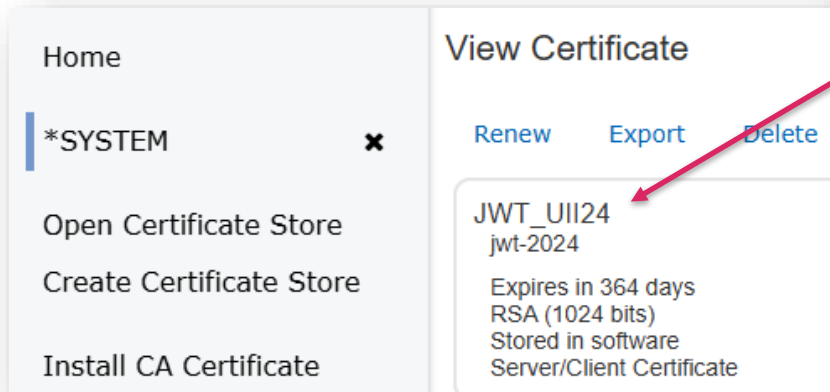
- Toujours dans `\www\{JWT_PRD}\wlp\usr\servers\{JWT_PRD}\server.xml`
 - id
 - Identifie le constructeur de JWT
 - On peut en faire plusieurs avec des configurations diverses
 - Fait partie de l'URL : `https://itest10:11002/jwt/ibm/api/uii24/token`

```
<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.gala.lan"/>
```

```
<keyStore id="defaultKeyStore"  
location="/QIBM/UserData/ICSS/Cert/Server/DEFAULT.KDB" password="{xor}Njo2bmZmbQ=="  
provider="IBMi50SJSSEProvider" type="IBMi50SKeyStore"/>
```

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>

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.
issuer	string		An Issuer is a case-sensitive URL using the HTTP or HTTPS scheme that contains scheme, host

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

Obtenir un jeton

- <https://itest10:11002/jwt/ibm/api/uii24/token>

- Produit

```
{"token":
```

```
"eyJraWQwIjoiJDRm9kVG5rSTNhUHM2UkYwR0owcU5ZU1RGYTdSM1NST3M1MlZERTdldkswIiwidHlwIjoilSlldUIiwiiYWxnIjoilU1MyNTYifQ.eyJ0b2t1b1l90eXB1IjoilQmVhcmVyiwiYXVkJpbInV1aTIwMjQiLCJucyIjZdWIIiOiJqd3QiLCJ1cG4iOiJqd3QiLCJyZWFSbSI6ImRlZmF1bHRSZWFsbSIsImR0aSI6InJPS2IwU2l1YnZUM0tIc3EiLCJpc3MiOiJodHRwczovL2p3dC5nYWlhLmxhbiIsImV4cCI6MTczMTYyNjE2MywiaWF0IjoixNzmxNjI2MTAzfQ.PEiyTOdiDdY1d1Qds4mckiWvnRCwy6-EGd1HEcaJYmQMmF5kUALqRdeUbkf2DGQZDmY_BNdf9TYHu_uKT878PbiV3aVK0448iEzI3xuNWqhQnNDlbnGqmYJtyuuubOMOljq6PU_YmvB3wqEiY7ICT4JLq3R0SPy5mvPhC2fLh3Mrx0tRiyRahUZqT4uJ9qpFV_0gDR6RePYozkb1LiYSKwqu1KCCXgxtMXcuhM1Lo8zcAdEMFkXz-OE3NN-IobIHbqti6pLaalKIh-F0XFayPgL_ER0wtDoGI3Bltswu0UETm4VLF10Jd450iyuHh83pYCGAsOZRKpaGobwDb4U8jA"}
```


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="uui24"
  keyAlias="JWT_UII24" keyStoreRef="defaultKeyStore"
  expiresInSeconds="3600s"
  audiences="uui2024,nb"
  jti="true"
  issuer="https://jwt.gaia.lan"/>
```

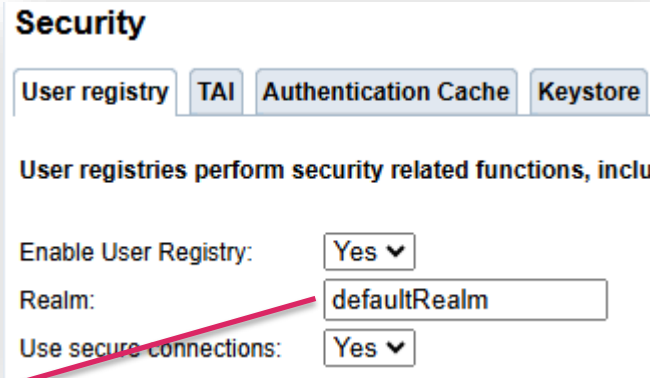
aud : audiences dans fichier de conf
sub & upn : utilisateur (jwt dans notre exemple)
jti : identifiant unique de jeton (pour jeton à usage unique par exemple)
iss : issuer
exp : expiration (secondes depuis EPOCH)
iat : date de création (secondes depuis EPOCH)

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": "eyJraWQiOiJDRm9kVG5rSTNhUHM2UkYwR0owcU5ZU1RGTdSM1NST3M1M1ZERTdldkswIiwidHlwIjoiSldUIiwiaWYwXn..."
```

- 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.



Authentication via JWT avec IWS

Notre service de test

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

Redeploy Service

Specify security constraint - Step 2 of 8

The security constraint limits who can access the web service. ?

Secure transport required: Yes ▾

Protect using authentication method: *NONE ▾

Specify SQL statements that will be externalized as a Web service: ?

	Procedure name	SQL statement/Parameter name
<input type="checkbox"/>	listeALLOBJ	<pre>SELECT AUTHORIZATION_NAME, STATUS, NO_PASSWORD_IN TEXT_DESCRIPTION FROM QSYS2.USER_INFO WHERE SPECIAL_AUTHORITIES LIKE '%*ALLOBJ%' OR AUTHORIZATION_NAME IN (SELECT USER_PROFILE_NAME FROM QSYS2.GROUP_PROFILE_ENTRIES WHERE GROUP_PROFILE_NAME IN (SELECT AUTHORIZATION_NAME FROM QSYS2.USER_INFO WHERE SPECIAL_AUTHORITIES like '%*ALLOBJ%')) ORDER BY AUTHORIZATION_NAME</pre>

Specify User ID for this Service: ?

Use server's user ID

Specify an existing user ID

User ID:

Update the server's user ID to have *USE authority to this user ID.

Use authenticated user ID

Service Properties

General Security Methods Swagger Connection Pool JDBC Properties

Service information ?

Resource Name: rootUsers

Resource description: Profils ALLOBJ

URI path template: /

Startup type: Automatic ▾

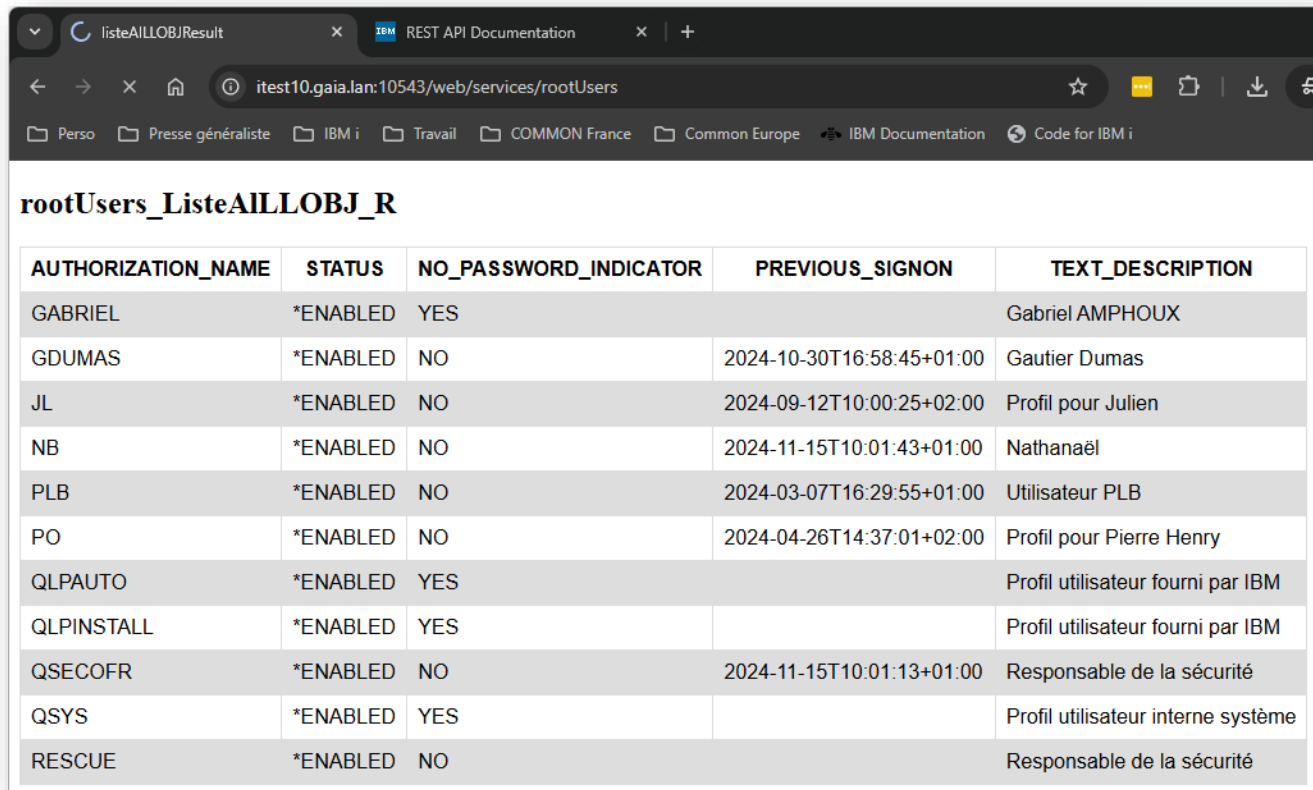
Service install path: /www/JWT_CNS/webservices/services/rootUsers

Base resource URL: https://itest10.gaia.lan:10543/web/services/rootUsers

Update the server's user ID to have *USE authority to this user ID.

Notre service de test

- Produit
- A sécuriser !



The screenshot shows a web browser window with the URL `itest10.gaia.lan:10543/web/services/rootUsers`. The page title is `rootUsers_ListeAILLOBJ_R`. The table below lists various user profiles with their authorization names, status, password indicators, previous signon times, and descriptions.

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 : authentication

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

Security

User registry TAI Authentication Cache Keystore CORS HSTS

User registries perform security related functions, including authentication and authorization, for app

Enable User Registry: Yes

Realm: defaultRealm

Use secure connections: No

Path to role definition file: /www/JWT_CNS/conf/roles.xml

Path to group definition file: /www/JWT_CNS/conf/groupees.xml

Normalize login name case: *NONE

User registry:

Use user profiles for authentication

NOTE: The server user ID must have *USE authority to user

Use validation list for authentication

Use file for authentication

```
www > JWT_CNS > conf > roles.xml
1 <?xml version="1.0" encoding="UTF-8"?>
2 <server>
3   <IBMiUserRegistry>
4     <role name="admin">
5       <user name="nb" />
6     </role>
7     <role name="restreint">
8       <user name="jwtuserrst" />
9       <group name="partenaire" />
10    </role>
11    <role name="all">
12      <user name="jwtuserall" />
13      <group name="gaia" />
14    </role>
15  </IBMiUserRegistry>
16 </server>
```

```
www > JWT_CNS > conf > users.xml
1 <?xml version="1.0" encoding="UTF-8"?>
2 <server>
3   <IBMiUserRegistry>
4     <user name="jwtuserrst" password="jwtuserrst"/>
5     <user name="jwtuserall" password="jwtuserall"/>
6     <user name="nb" password="pwdnb"/>
7     <user name="jl" password="pwdjl"/>
8     <user name="plb" password="pwdplb"/>
9     <user name="jms" password="{xor}Lz4sLChvLTs"/>
10  </IBMiUserRegistry>
11 </server>
```

```
www > JWT_CNS > conf > groupees.xml
1 <?xml version="1.0" encoding="UTF-8"?>
2 <server>
3   <IBMiUserRegistry>
4     <group name="gaia">
5       <member name="nb" />
6       <member name="plb" />
7       <member name="jl" />
8       <member name="jwtuser" />
9     </group>
10    <group name="partenaire">
11      <member name="gabriel" />
12      <member name="gd" />
13    </group>
14  </IBMiUserRegistry>
15 </server>
```


Etape 1 : authentication

- Au niveau du service

Redeploy Service

Specify security constraint - Step 2 of 8

The security constraint limits who can access the web service. ?

Secure transport required: Yes ▾

Protect using authentication method: *BASIC ▾

All available roles

admin
restreint
all



Authorized roles

admin
all

Profil authentifié ↔ Profil exécution

Redeploy Service

Specify User ID for this Service - Step 6 of 8

The service requires an IBM i user ID to run the Web service business logic. T

Specify User ID for this Service: ?

Use server's user ID

Specify an existing user ID

User ID: NB

Update the server's user ID to have *USE authority to this user ID.

Use authenticated user ID

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

- Appel du service

- Utilisateurs

- NB : OK
- JWTUSERRST : ERREUR
- JWTUSERALL : OK

The screenshot shows a login dialog box with the title "Se connecter" and the URL "https://itest10.gaia.lan:10443". It contains two input fields: "Nom d'utilisateur" and "Mot de passe". Below the fields are two buttons: "Se connecter" (blue) and "Annuler" (white with blue border). A red arrow points from the text "Casse en fonction de Normalize" to the "Nom d'utilisateur" field.

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
```


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

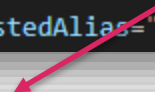
Etape 2 : authentication via JWT

- 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>IPMillsonRegistry-1.0</feature>
  <feature>jwt-1.0</feature>
</featureManager>
```

```
<jwtConsumer id="jwtcons"
  audiences="uui2024,nb"
  issuer="https://jwt.gaia.lan"
  trustStoreRef="defaultKeyStore"
  trustedAlias="JWT_UII24"/>
```

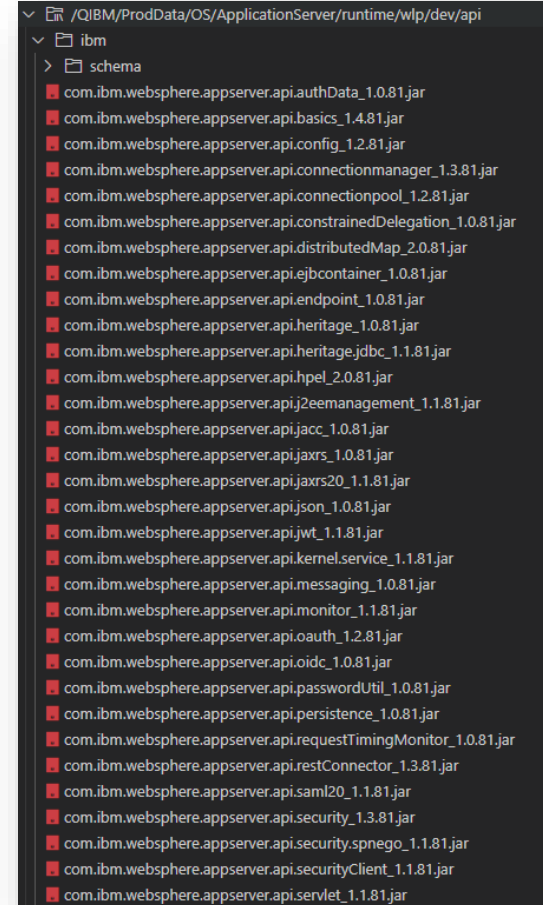
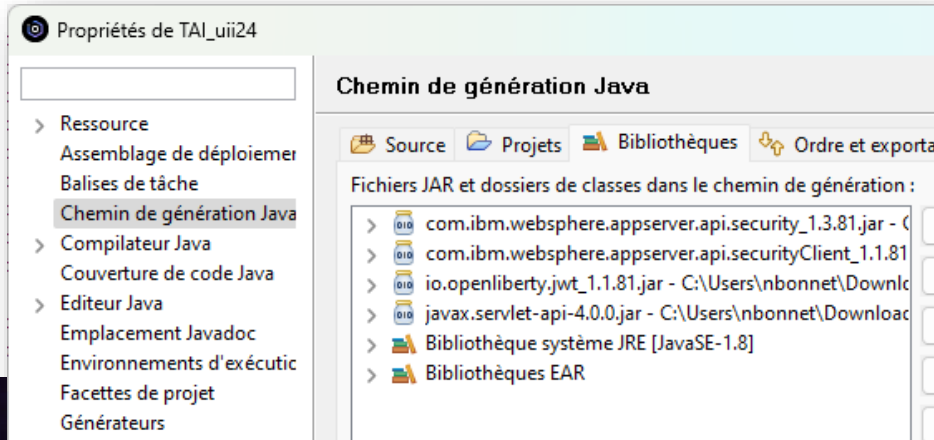
```
<keyStore id="defaultKeyStore"
  location="/QIBM/UserData/ICSS/Cert/Server/DEFAULT.KDB"
  password="{xor}Njo2bmZmbQ=="
  provider="IBMi50SJSSEProvider"
  type="IBMi50SKeyStore"/>
```



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 : authentication via JWT

- Code
 - Authentication = 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 : authentication 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 : authentication via JWT

- Exporter le .jar puis

[JWT_CNS](#) > Security

Security

User registry TAI Authentication Cache Keystore CORS HSTS

The trust association interface (TAI) is a service provider API that enables the integration of third-party security services. ?

Enable TAI: Yes ▾

Interceptor class name: fr.gaia.tai.gaiaTAI e.g. com.ibm.sample.SimpleTAI

Invoke TAI for unprotected applications: No ▾

Allow fallback to application authentication: No ▾

Libraries for interceptor:

	Library
Example	/myjars/simpletai.jar
Example	/myjars/*.*jar
<input type="radio"/>	/www/JWT_CNS/conf/gaiaTAI.jar

Add Remove All

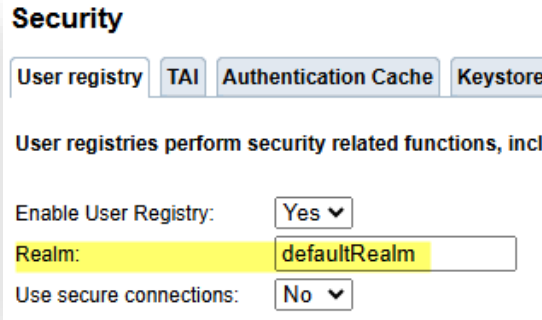
Etape 2 : authentication via JWT

■ 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="uui24"  
  keyAlias="JWT_UUI24" keyStoreRef="defaultKeyStore"  
  expiresInSeconds="3600s"  
  audiences="uui2024,nb"  
  jti="true"  
  issuer="https://jwt.gaia.lan"/>
```

```
<jwtConsumer id="jwtcons"  
  audiences="uui2024,nb"  
  issuer="https://jwt.gaia.lan"  
  trustStoreRef="defaultKeyStore"  
  trustedAlias="JWT_UUI24"/>
```



Avec du code

- Par SQL

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

00001

eyJraWQiOiJwbmXnQ2RRazg5TkRfWjNXcUlzdThqeXpSN0hScURuSWlwdldnTnlSc0ZrIiwidHlwIjoiSldUIiwiaWF0IjoiYXNjaWYyYXVkaWpMjAyNCIsInN1YiI

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

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/>

MEREC

The word "MEREC" is displayed in large, white, 3D block letters with a soft drop shadow. Each letter contains a different professional photograph: 'M' shows a woman in a green top; 'E' shows a man in a green patterned shirt; 'R' shows a woman in a light blue top with her hands clasped; 'E' shows a man in a blue suit and yellow tie; 'C' shows a woman in a blue top and glasses.