OMPI
ORGANISATION MONDIALE
DE LA **PROPRIÉTÉ**
INTELLECTUELLE

1er octobre 2021

SERVICES WEB DU SYSTÈME DE LA HAYE

Table des matières

INTRODUCTION

PORTÉE DU DOCUMENT

Le présent document présente les services Web du système de La Haye, interface d'échange de données machine à machine pour le système de La Haye.

GÉNÉRALITÉS

Les services Web du système de La Haye sont un protocole fondé sur une API HTTPS/REST, fiable et hautement disponible, pour l'échange de données avec le système de La Haye. Ils peuvent être utilisés pour envoyer ou recevoir des données.

Les services Web du système de La Haye peuvent être utilisés pour :

- envoyer des décisions ou des demandes indirectes;
- vérifier le statut de l'importation;
- connaître le Statut du traitement;
- récupérer des Bulletins des dessins et modèles internationaux de l'OMPI;
- récupérer des copies confidentielles (Offices de propriété intellectuelle procédant à l'examen uniquement).

Les services Web du système de La Haye sont le moyen privilégié pour l'échange de données relatives au système de La Haye. Les Offices de propriété intellectuelle sont donc vivement encouragés à utiliser ces services dès le début. Les Offices qui échangent déjà des données avec le système de La Haye par EDI, sur papier ou par d'autres canaux sont encouragés à migrer vers les services Web du système de La Haye.

AUTHENTIFICATION ET SÉCURITÉ

SÉCURITÉ DE L'API

L'API des services Web du système de La Haye est conçue pour communiquer des données confidentielles de machine à machine.

L'authentification est basée sur une signature à clé asymétrique qui fait partie du Financial-grade API Security Profile 1.0. Ce dernier peut être appliqué aux API de n'importe quel secteur du marché nécessitant un niveau de sécurité supérieur à celui fourni par les normes OAuth ou OpenID Connect. Cela signifie qu'il offre un profil de sécurité avancé par rapport à OAuth, adapté à la protection des API présentant un risque inhérent élevé.

PRODUCTION DE PAIRES DE CLÉS ET IDENTIFIANTS CLIENTS

Le schéma ci-dessous montre le processus de bout en bout pour enregistrer un identifiant client et une clé publique auprès de l'OMPI, ainsi que l'adresse IP publique de l'application client.

Mesures à prendre par l'Office

1. Produire une paire de clés publique et privée (voir l'Appendice A : Client pour production de paires de clés openSSL).

2. Produire le certificat x509 au moyen de la clé publique.

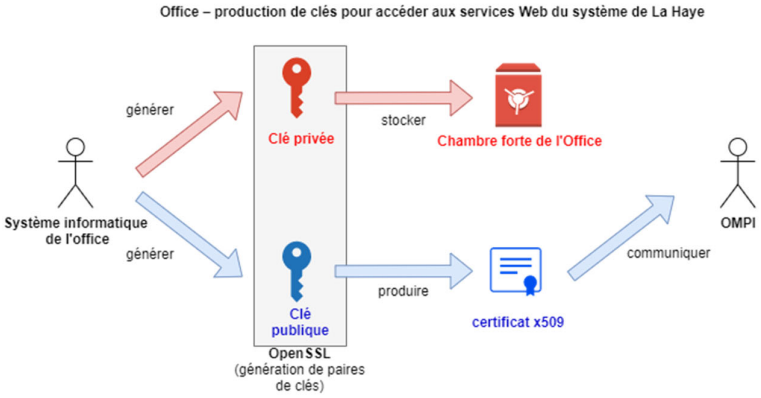
3. Demander l'accès aux services Web du système de La Haye en envoyant un message électronique à hague.it@wipo.int accompagné

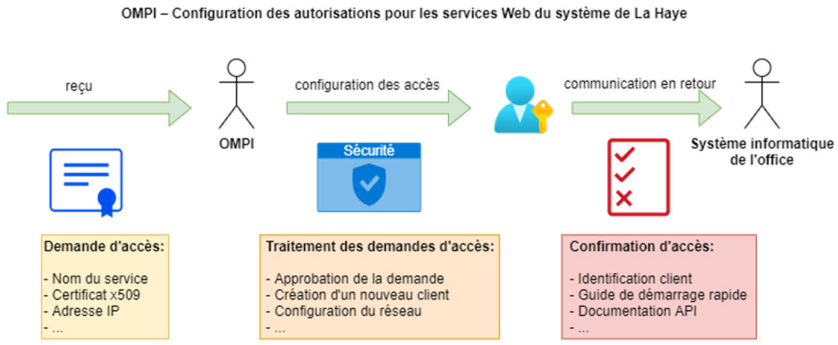
   a) du formulaire de l'OMPI dûment rempli (voir l'Appendice B : Formulaire de demande d'accès à l'API du Département des technologies de l'information et de la communication de l'OMPI); et

   b) du certificat x509.

Mesures à prendre par l'OMPI :

1. Après réception des pièces susmentionnées, produire l'identifiant client.
2. Attribuer/associer la clé publique à l'identifiant client.
3. Enregistrer l'adresse IP sur une liste blanche.
4. Configurer les services Web du système de La Haye pour autoriser les requêtes par l'identifiant client.
5. Confirmez l'identifiant client à l'Office de propriété intellectuelle.





AUTHENTIFICATION AUPRÈS DES SERVICES WEB DU SYSTÈME DE LA HAYE

Une fois que l'identifiant client, la clé publique et l'adresse IP publique sont enregistrés auprès de l'OMPI et que les services Web du système de La Haye ont été configurés, l'Office de propriété intellectuelle peut commencer à utiliser l'API.

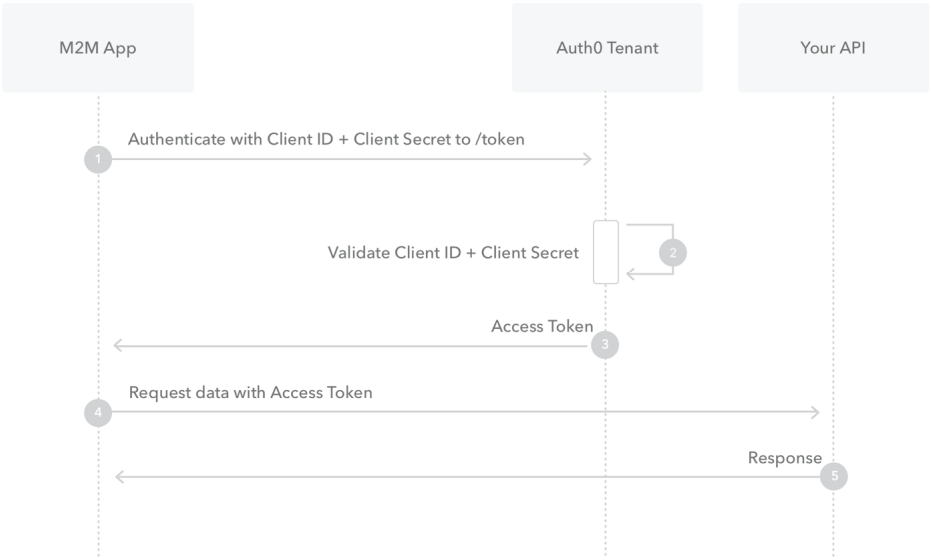
Le schéma ci-dessous illustre cette interaction :

1. Requête HTTPS à l'entité Auth0 avec l'identifiant client et le jeton JWT signé au moyen de la clé privée. Remarque : la requête **doit** provenir de l'IP figurant sur la liste blanche.

2. La requête HTTPS est validée et le jeton d'accès JWT est produit.

3. Si tout est en ordre, le jeton d'accès JWT est renvoyé avec une durée de validité d'une heure.

4. Les requêtes HTTPS ultérieures pendant le délai imparti peuvent être effectuées au moyen du même jeton d'accès JWT.



Traduction de l'image ci-dessus :

M2M App	Application M2M (machine à machine)
Auth0 Tenant	Entité Auth0
Your API	Votre API
Authenticate with Client ID + Client Secret to /token	Authentification avec l'identifiant client + jeton
Validate Client ID + Client Secret	Validation de l'identifiant client + jeton
Access Token	Jeton d'accès
Request data with Access Token	Requête de données avec jeton d'accès
Response	Réponse

DESCRIPTION DE L'API

L'API des services Web du système de La Haye met en œuvre les services REST suivants :

1. Envoi des demandes indirectes et décisions au système de La Haye (POST /request).

2. Vérification du statut de l'importation pour une demande indirecte ou une décision envoyée (GET /request/import).

3. Interrogation de l'état d'une *service request* (SR)[1] particulière (GET /request/(serviceRequestId().

4. Récupération d'un Bulletin (GET /publication/bulletin/(weekId().

5. Récupération d'une copie confidentielle (GET /publication/copy/confidential/(weekId().

On trouvera les détails complets sur l'API des services Web du système de La Haye (paramètres, réponses, etc.) à l'AppendiceAPPENDICE D : .

Tous le contenu utile de (requêtes et réponses) sont basées sur la norme XML utilisée à l'OMPI, à savoir la norme ST.96. Des détails complets sur la norme ST.96 v4.0 et les XSD sont disponibles à l'adresse https://www.wipo.int/standards/fr/st96/v4-0/. Des extensions mineures spécifiquement requises pour les services Web sont en cours de normalisation.

Remarque : il existe un service appelé pingMe qui peut être utilisé pour vérifier la connectivité entre le client et les services Web du système de La Haye. Il n'a aucune fonction, mais il est mis à disposition à des fins de tests techniques et de validation.

Envoi de demandes indirectes et de décisions au système de La Haye (POST /request)

L'envoi de demandes et de décisions au système de La Haye se fait par l'intermédiaire d'une requête de type POST, dont le contenu utile est le fichier d'importation (voir ci-dessous).

Un identifiant d'importation est renvoyé pour chaque fichier d'importation si tout est en ordre, signifiant que ledit fichier a été reçu et qu'il sera traité par le Bureau international.

Cet identifiant d'importation peut ensuite être utilisé pour récupérer le numéro de SR (GET request/import), qui à son tour peut être utilisé pour récupérer le statut de la SR (GET request).

Le contenu utile d'une demande indirecte de protection et d'une décision est un fichier ZIP unique contenant les documents (PDF) et les images (JPEG), ainsi que la description des données bibliographiques au format XML selon la norme ST.96.

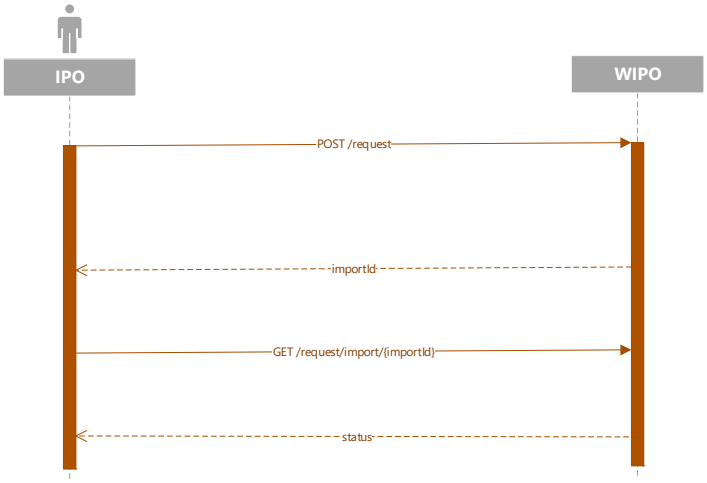
- Ces documents et images doivent être situés dans le fichier ZIP au chemin relatif indiqué dans le XML.

- Un fichier ZIP (donc un contenu utile de requête) ne doit contenir qu'une seule demande indirecte ou une seule décision.

- Un contenu utile ne peut contenir plus d'un fichier XML.

- Des échantillons sont disponibles à l'adresse ftp://ftpird.wipo.int/ST96_V_4_0_test/import-packages-4.0.zip.

[1] Au sein du système de La Haye, transaction correspondant à une demande de protection ou à tout changement ou opération ultérieure sur cette demande (par exemple, une décision d'octroi ou de refus de protection).



Interrogation du statut d'une SR particulière (GET request/{serviceRequestId})

Après l'importation du fichier dans le système de La Haye, une transaction est appelée *service request* (SR) et se voit attribuer un numéro (SR number). Ce numéro de SR peut être récupéré par l'intermédiaire du service GET request/import (voir ci-dessus).

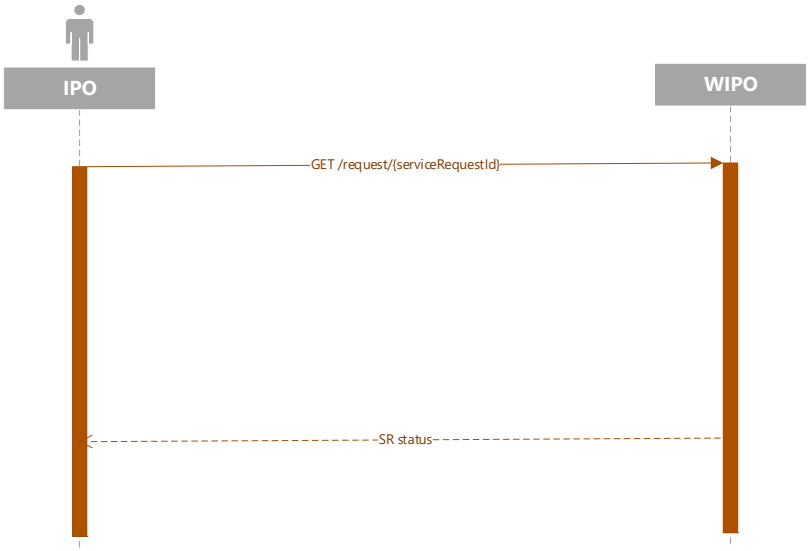
Une fois que le numéro de SR est disponible, l'état de la SR peut être récupéré au moyen du *service request*.

Les types de statut sont :

- indéfini;
- en traitement;
- en attente de régularisation;
- enregistré;
- abandonné;
- radié.

En réponse à l'interrogation de l'état d'une SR, les services Web du système La Haye renvoient un contenu utile conformément à la norme ST.96 comprenant :

- l'identifiant de la requête;
- le statut du traitement;
- le numéro de la SR;
- l'IRN (numéro d'enregistrement international);
- le cas échéant, la date de publication prévue.



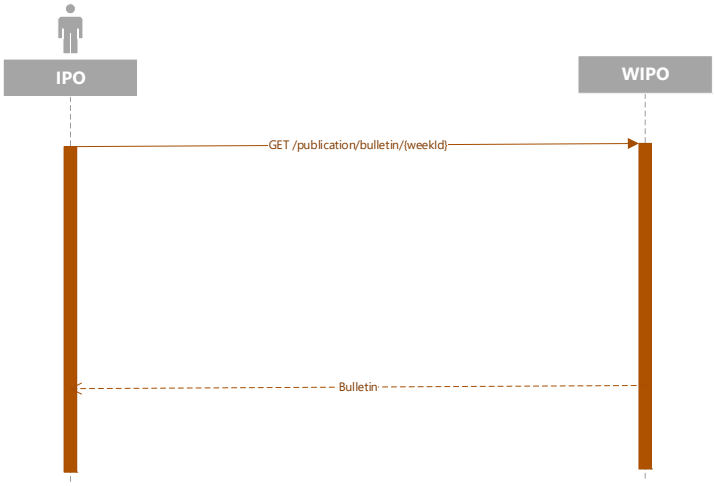
Récupération d'un Bulletin (GET /publication/bulletin/{weekId})

Les bulletins sont publiés par le Bureau international chaque semaine, généralement le vendredi en milieu de journée (heure d'Europe centrale). Ceux-ci peuvent être demandés à tout moment dès lors qu'ils ont été produits. Le format du paramètre weekId est yyyyww.

Le contenu utile de la réponse est un fichier ZIP contenant :

- les données bibliographiques du Bulletin sous forme de fichier conforme à la norme ST.96;
- des dossiers d'images correspondant aux enregistrements ou aux corrections d'images inclus.

À titre d'exemple, on trouvera les bulletins (les copies confidentielles ont la même architecture) à l'adresse ftp://ftpird.wipo.int /ST96_V_4_0.



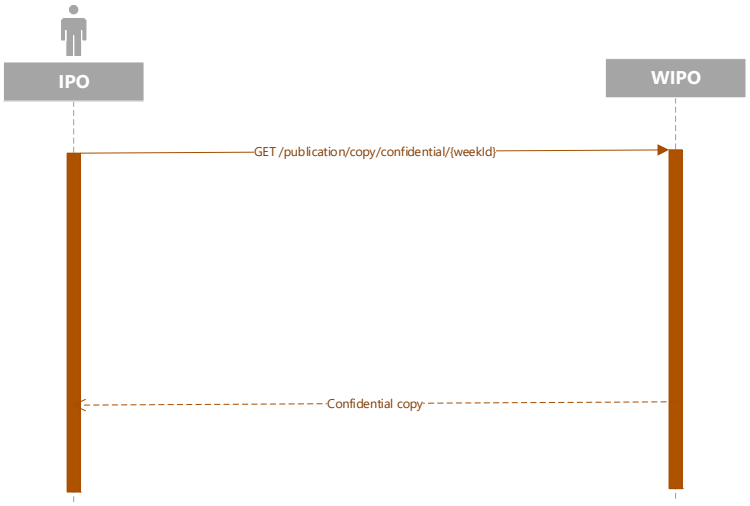
Récupération d'une copie confidentielle (GET /publication/copy/confidential/{weekId})

Les copies confidentielles ont la même architecture que les Bulletins.

Les copies confidentielles sont publiées par le Bureau international chaque semaine, généralement le vendredi en milieu de journée (heure d'Europe centrale). Celles-ci peuvent être récupérées à tout moment dès lors qu'elles ont été produites. Le format du paramètre weekId est yyyyww.

Le contenu utile de la réponse est un fichier ZIP contenant :

- les données bibliographiques de la copie confidentielle sous forme de fichier conforme à la norme ST.96;
- des dossiers d'images correspondant aux enregistrements ou aux corrections d'images inclus.



APPENDICE A : CLIENT POUR PRODUCTION DE PAIRES DE CLÉS OPENSSL

Produire une paire de clés asymétriques privées/publiques et un certificat x509 pour l'enregistrement OMPI de machine à machine.

Production d'artéfacts pour l'inscription à l'OIDC de l'OMPI

```
#!/bin/bash

# Set the environment
PRIVATE_KEY_ES256=hague4offices_private.pem
PUBLIC_KEY_ES256=hague4offices_public.pem
CLIENT_NAME=DAS

# Generates the ES256 keys
openssl ecparam -genkey -name prime256v1 -noout -out "${PRIVATE_KEY_ES256}"

# Extracts the public key
openssl ec -in "${PRIVATE_KEY_ES256}" -pubout -out "${PUBLIC_KEY_ES256}"

# Generates an x509 certificate
CERT_KEY_ES256=es256_cert.pem
OPENSSL_CONF=./openssl.cnf
CERT_CN="${CLIENT_NAME} private_key_jwt authentication"

# Build the certificate config file
printf '[ req ]\n' > "${OPENSSL_CONF}"
printf 'prompt = no\n' >> "${OPENSSL_CONF}"
printf 'distinguished_name = req_distinguished_name\n' >> "${OPENSSL_CONF}"
printf '[ req_distinguished_name ]\n' >> "${OPENSSL_CONF}"
printf 'CN = %s\n' "${CERT_CN}" >> "${OPENSSL_CONF}"

# Creates the x509 certificate
openssl req -x509 -new -config "${OPENSSL_CONF}" -key "${PRIVATE_KEY_ES256}" -out
"${CERT_KEY_ES256}"
```

1. Envoyer **es256_cert.pem** à l'OMPI pour la configuration de l'accès aux services Web du système de La Haye (**hague4offices_private.pem** doit toujours rester secret et ne jamais être communiqué).

2. Attendre que l'OMPI communique en retour **l'identifiant client** et la **portée** après la configuration.

3. Tester la communication au moyen de l'application client de test fournie par le système de La Haye (lien à confirmer).

APPENDICE B : FORMULAIRE DE DEMANDE D'ACCÈS À L'API DU DÉPARTEMENT DES TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION DE L'OMPI

Les formulaires ci-dessous sont des projets de formulaires de l'OMPI. Veuillez noter que la version finale de ce formulaire est en attente et sera confirmée (01/09/2021).

Veuillez remplir ce formulaire pour fournir des informations générales sur le contexte.

Informations générales	
Type de demande?	Demande de création ☐ Demande de mise à jour ☐
Décrire les informations mises à jour[2]	Coordonnées ☐ Plage d'IP/IP du client ☐ Certificat ☐ Portées ☐
Environnement[3]	Production
Nom de l'application API	
Description de l'application	
URL de l'API	
Sponsor de l'application	
Adresse électronique du sponsor de l'application[4]	
Nom du technicien chargé de l'application	
Adresse électronique du technicien chargé de l'application[5]	
Qui aura accès à l'application?	Personnel interne ☐ Personnes externes ☐
Comment l'API est-elle protégée?	Utilisation d'un jeton d'accès acquis via le flux d'informations d'identification du client Oauth 2

2 Veuillez fournir ces informations uniquement en cas de demande de mise à jour.
3 Veuillez sélectionner un environnement.
4 Sera utilisée à des fins de notification lorsque le déploiement d'un composant Oauth2 Provider est prévu en production et pourrait avoir un impact sur l'application.
5 Sera utilisée à des fins de notification lorsque le déploiement d'un composant Oauth2 Provider est prévu en production et pourrait avoir un impact sur l'application.

Veuillez remplir ce formulaire pour fournir des informations sur le client :

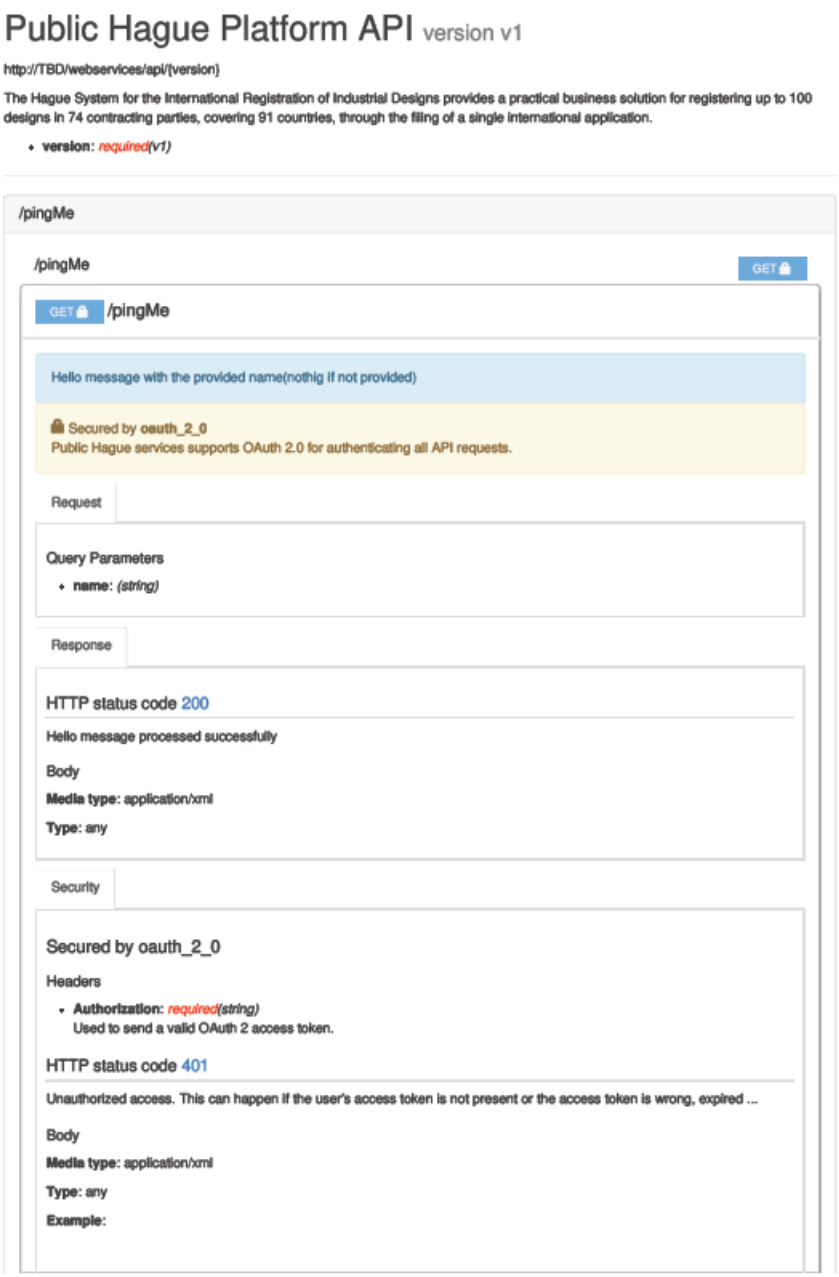
Protection des API à l'aide d'OAuth2	
Identifiant client	Sera fourni par l'OMPI
Type de client	Confidentiel
Portées prises en charge (facultatif)	Profil par défaut
CERTIFICAT (X509V3 – ES256)	
PLAGE IP/IP DU CLIENT	
Méthode d'authentification du client	private_key_jwt (le client envoie ses informations d'identification sous forme de JWT)

APPENDICE C : SNIPPET POUR OBTENIR UN JETON D'ACCÈS À PARTIR DE WIPO DEV OPENAM

Le script bash ci-dessous est un exemple de demande d'authentification OMPI au moyen de la clé privée de l'office de propriété intellectuelle :

```
#!/bin/bash
PRIVATE_KEY_ES256=es256_private.pem
CLIENT_ID=das-api-auth
SCOPE="das-api/das-access"
ISSUER="https://logindev.wipo.int/am/oauth2"

# https://logindev.wipo.int/am/oauth2/.well-known/openid-configuration
OIDC_CONFIG_JSON=$(curl -k "${ISSUER}/.well-known/openid-configuration")

# Generic way to obtain the token endpoint
TOKEN_ENDPOINT=$(printf '%s' ${OIDC_CONFIG_JSON} | jq -r ".token_endpoint")
UTC_TIME=$(date -u +%s)
EXP_TIME=$(expr "$UTC_TIME" + 10)
JWT_ID=Un1qu3i0

JSON='{'
JSON=${JSON}$(printf '"iss":"%s"' ${CLIENT_ID})
JSON=${JSON}$(printf ,"sub":"%s"' ${CLIENT_ID})
JSON=${JSON}$(printf ,"aud":"%s"' ${TOKEN_ENDPOINT})
JSON=${JSON}$(printf ,"exp":%s' ${EXP_TIME})
JSON=${JSON}'}'

JSON_HEADER_B64=$(printf '{"alg":"ES256","typ":"JWT"}' | jq -cj | base64 -w0 | tr -d
'\n=' | tr '+/' '-_')

JSON_PAYLOAD_B64=$(printf $JSON | jq -cj | base64 -w0 | tr -d '\n=' | tr '+/' '-_')

JSON_SIGNATURE_ASN1_B64=$(printf '%s.%s' $JSON_HEADER_B64 $JSON_PAYLOAD_B64 | openssl
dgst -sha256 -sign"${PRIVATE_KEY_ES256}" | openssl asn1parse -inform DER | base64 -w0)
JSON_SIGNATURE_HEX=$(printf $JSON_SIGNATURE_ASN1_B64 | base64 -d | sed -n '/INTEGER/p'
| sed 's/.*INTEGER\s*://g' | sed -z 's/[^0-9A-F]//g')
JSON_SIGNATURE_B64=$(printf $JSON_SIGNATURE_HEX | xxd -p -r | base64 -w0 | tr -d '\n='
| tr '+/' '-_')

JWT_ASSERTION=$(printf '%s.%s.%s' $JSON_HEADER_B64 $JSON_PAYLOAD_B64
$JSON_SIGNATURE_B64)

# echo $JWT_ASSERTION
# Access token private_key_jwt
# --insecure is only needed when testing within WIPO premises (because of the
proxy...)
curl \
  --header "Content-Type: application/x-www-form-urlencoded" \
  --data-urlencode "grant_type=client_credentials" \
  --data-urlencode scope="${SCOPE}" \
  --data-urlencode "client_assertion_type=urn:ietf:params:oauth:client-assertion-
type:jwt-bearer" \
  --data-urlencode "client_assertion=${JWT_ASSERTION}" \
  --url "${TOKEN_ENDPOINT}"
```

APPENDICE D : API DE LA PLATEFORME PUBLIQUE DU SYSTÈME DE LA HAYE



Public Hague Platform API version v1

http://TBD/webservices/api/{version}

The Hague System for the International Registration of Industrial Designs provides a practical business solution for registering up to 100 designs in 74 contracting parties, covering 91 countries, through the filing of a single international application.

- **version:** *required(v1)*

/pingMe

/pingMe GET

GET /pingMe

Hello message with the provided name(nothig if not provided)

Secured by **oauth_2_0**
Public Hague services supports OAuth 2.0 for authenticating all API requests.

Request

Query Parameters

- **name:** *(string)*

Response

HTTP status code 200

Hello message processed successfully

Body

Media type: application/xml

Type: any

Security

Secured by oauth_2_0

Headers

- **Authorization:** *required(string)*
  Used to send a valid OAuth 2 access token.

HTTP status code 401

Unauthorized access. This can happen if the user's access token is not present or the access token is wrong, expired ...

Body

Media type: application/xml

Type: any

Example:

Remarque : des informations supplémentaires seront ajoutées aux versions ultérieures de ce document.