WIPO
WORLD
INTELLECTUAL PROPERTY
ORGANIZATION

October 1, 2021

HAGUE WEB SERVICES

CONTENTS

INTRODUCTION

SCOPE

This document is an introduction to the Hague Web Services (HWS), a machine-to-machine interface (M2M) to the Hague System (Hague).

GENERAL

HWS are a secure, highly-available, reliable, HTTPS/REST API-based protocol for exchanging data with the Hague System. They can be used for sending or receiving data.

HWS can be used to:

- Send decisions or indirect applications
- Check import status
- Query processing status
- Retrieve Hague Bulletins.
- Retrieve confidential copies (Examining IP Offices only).

HWS is the preferred Hague data exchange channel. IPOs are therefore strongly encouraged to use the HWS from the start. Offices already exchanging data with Hague via EDI/paper/other channels are encouraged to migrate to the HWS.

AUTHENTICATION AND SECURITY

API SECURITY

The HWS API is designed for machine-to-machine communication with confidential payloads.

The authentication is based on an asymmetric key signature that is part of the Financial-grade API Security Profile 1.0. The Financial-grade API security profile can be applied to APIs in any market area that requires a higher level of security than provided by standard OAuth or OpenID Connect.This means that it has an advanced security profile of OAuth that is suitable for protecting APIs with high inherent risk.

KEY PAIR GENERATION AND CLIENT ID PROVISIONING

The diagram below shows the end-to-end process to register an API client id and public key to WIPO as well as the public IP address of the client application.

Office Actions:

1. Generate a pair of public and private keys (see Appendix A: Client for OpenSSL key pair generation).

2. Generate the x509 certificate using the public key.

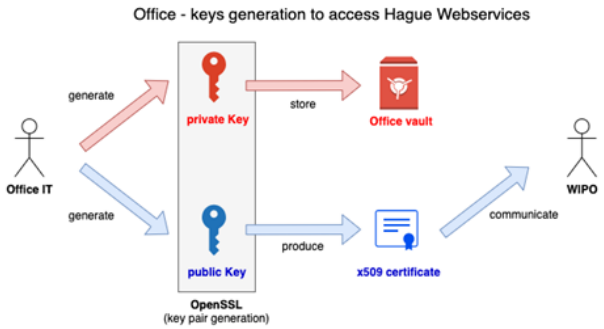
3. Request access to the HWS by sending an email to hague.it@wipo.int including:

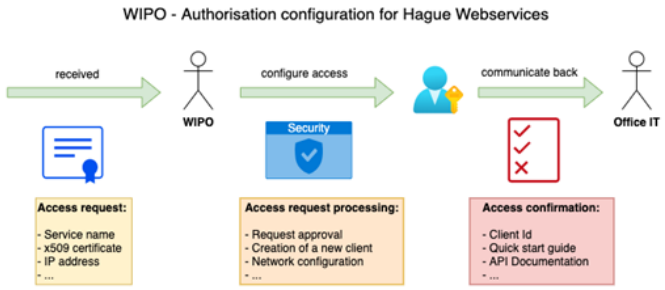
   (a) completed WIPO form (see Appendix B: APPENDIX B: WIPO ICTD API Request Access Form);

   (b) x509 certificate.

WIPO Actions:

1. After receipt of the above, generate the client ID.
2. Assign/link the public key to the client ID.
3. Whitelist the IP address.
4. Configure the HWS to authorize requests to the client ID.
5. Confirm the client ID to the IP Office.



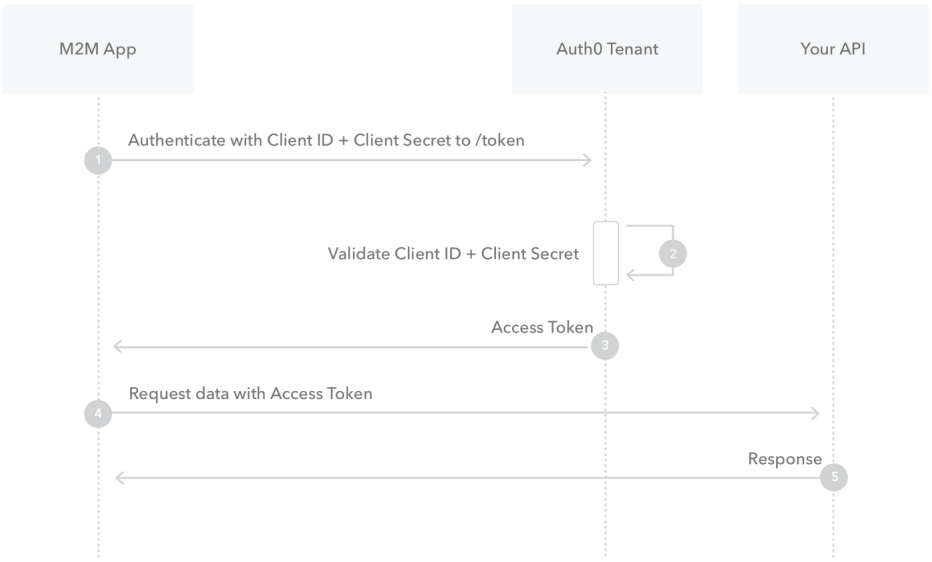


AUTHENTICATION TO HWS

Once the client ID, public key and public IP address is registered with WIPO, and the HWS services configured, the IP Office is ready to use the API.

The diagram below shows the interaction:

1. HTTPS request to the Auth0 tenant with client ID and JWT token signed by the private key. Note: the request **must** come from the whitelisted IP.
2. HTTPS request is validated and the JWT access token generated.
3. On success, the JWT access token is returned with a one-hour expiration.
4. Subsequent calls to the HTTPS endpoints within the expiration window can be performed using the same JWT access token.



API DESCRIPTION

The HWS API implements the following REST endpoints:

1. Sending indirect applications and decisions to Hague (POST /request).
2. Check import status for an indirect application or decision sent (GET /request/import) .
3. Querying the status of a particular Service Request (SR) (GET /request/{serviceRequestId}).
4. Retrieval of a Bulletin (GET /publication/bulletin/{weekId}).
5. Retrieval of a Confidential Copy (GET /publication/copy/confidential/{weekId}).

Full details about HWS API (parameters, responses, etc.) can be found in Appendix D: APPENDIX D: Public Hague Platform API.

All payloads are based on the XML standard in use at WIPO, namely ST.96. Full details about ST.96 v4.0, and the XSDs, can be found at https://www.wipo.int/standards/en/st96/v4-0/. Minor extensions specifically required for web services are in the process of being standardized.

Note: There is an endpoint called pingMe that can be used for checking connectivity between both the client side and HWS. It has no functionality, but is made available for technical testing and validation purposes.

Sending indirect applications and Decisions to Hague (POST /request)

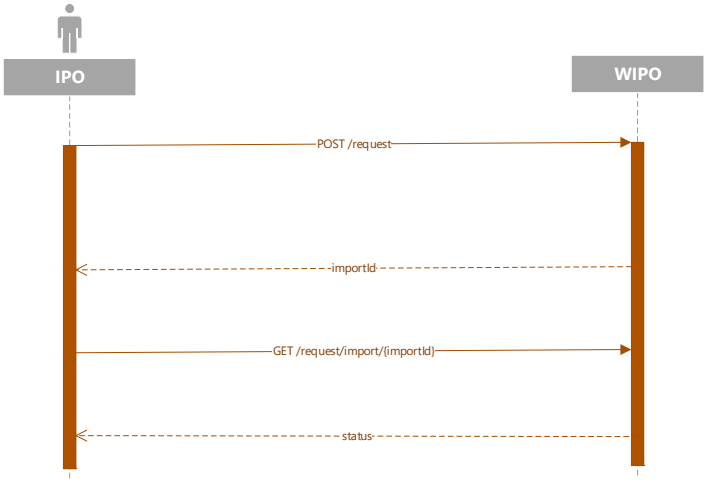
Sending applications and decisions to Hague is done through a POST request, where the payload is the import package (see below).

An import ID per package is returned on success, meaning that that the import package was received and will be processed by the IB.

This package import ID can later be used to retrieve the Service Request Number (GET request/import), and in turn the SRN can be used to retrieve the request status (GET request).

An application and decision request payload is a single ZIP file containing ST.96 XML and documents and images.

- These files must be located in the relative path as indicated in the XML.
- A ZIP file (therefore, a request payload) must contain only one application or one decision.
- A payload cannot contain more than one XML file.
- Samples can be found at ftp://ftpird.wipo.int/ST96_V_4_0_test/import-packages-4.0.zip.



Querying the status of a particular Service Request (GET request/{serviceRequestId})

After the package is imported in to the Hague system, a transaction is called a Service Request (SR) and is attributed an SRN (SR number). This SRN can be retrieved through the GET request/import endpoint (see above).

Once the SRN is available then the request status can be retrieved using the request endpoint.

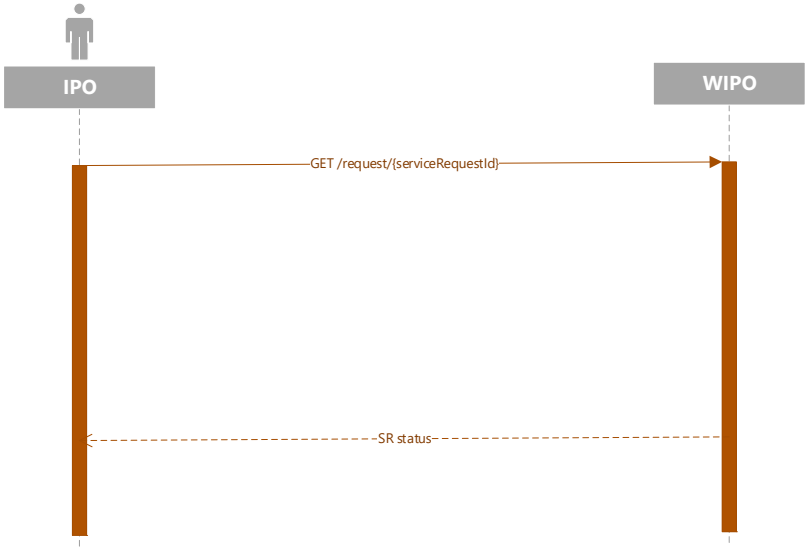
Status types are:

- Undefined
- Processing
- Pending Regularization
- Registered
- Abandoned
- Cancelled

As a response to querying the status of a SR, the Hague web services send back an ST.96 payload comprising:

- Request ID
- Processing status

- SRN
- IRN (International Registration Number)
- when relevant, Expected publication date.



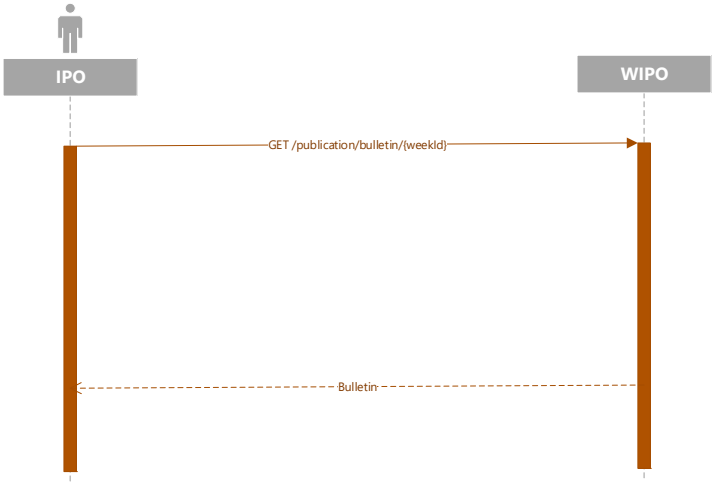
Retrieval of a Bulletin (GET /publication/bulletin/{weekId})

Bulletins are issued by the IB weekly, typically on a Friday night (Central European Time). These can be requested at any time from the moment they are generated. Where the weekId parameter format is yyyyww.

The response payload contains a ZIP file with the following contents:

- the Bulletin bibliographic data as an ST.96 file;
- folders of images corresponding to the included registrations or image corrections.

As an example, Bulletins payloads (confidential copies have the same architecture) can be found at ftp://ftpird.wipo.int /ST96_V_4_0.



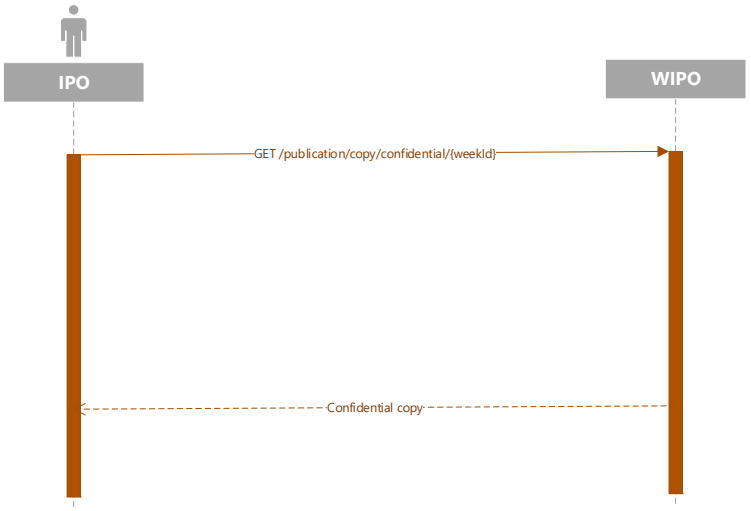
Retrieval of a Confidential Copy (GET /publication/copy/confidential/{weekId})

Confidential copies have the same architecture as Bulletins.

Confidential copies are issued by the IB weekly, typically on a Friday night (Central European Time). These can be requested at any time from the moment they are generated. Where the weekId parameter format is yyyyww.

The response payload contains a ZIP file with the following contents:

- the confidential copy bibliographic data as an ST.96 file;
- folders of images corresponding to the included registrations or image corrections.



APPENDIX A: CLIENT FOR OPENSSL KEY PAIR GENERATION

Generate a private / public asymmetric key pair and a x509 certificate for WIPO machine-to-machine registration.

Generate artifacts for WIPO OIDC enrolment

```
#!/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. Send **es256_cert.pem** to WIPO for Hague Webservices access configuration (**hague4offices_private.pem** should always be kept secret and never shared).
2. Wait for the **Client Id** and **scope** communicated back by WIPO after the configuration.
3. Test the communication using the Hague-provided test client application (link to be confirmed).

APPENDIX B: WIPO ICTD API REQUEST ACCESS FORM

The forms below are draft forms from WIPO. Please note that the final version of this form is pending and will be confirmed (01/09/2021).

Please fill in this form to provide general information about the context.

General information	
Request Type?	Creation request ☐ Update request ☐
Describe the updated information[1]	Contact information ☐ Client IP/IP range ☐ Certificate ☐ Scopes ☐
Environment[2]	Production
API Application Name	
Application Description	
API URL(s)	
Application Business Owner	
Application Business Owner Mail[3]	
Application Technical Contact Name	
Application Technical Contact Email[4]	
Who will access the application?	Internal people ☐ External people ☐
How Api is protected?	Using Access Token acquired via Oauth 2 Client Credentials flow

1 Please provide this information only in case of an update request.
2 Please select an environment.
3 Will be used for notification when a deployment of an Oauth2 Provider component is planned in production and could have an impact on the application.
4 Will be used for notification when a deployment of an Oatuh2 provider component is planned in production and could have an impact on the application.

Please fill in this form to provide information about the Client:

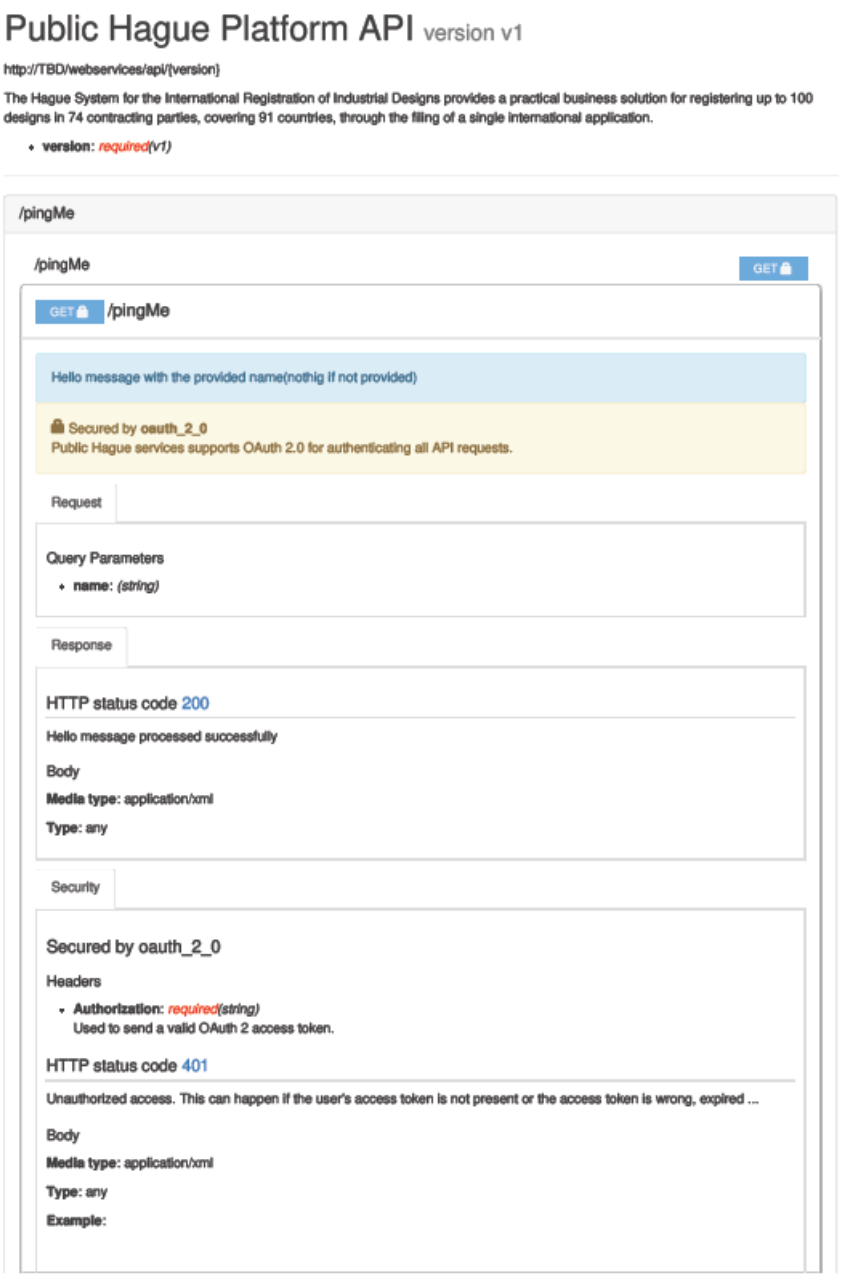
API protection using OAuth2	
Client ID	Will be provided by WIPO
Client Type	Confidential
Supported scopes (optional)	By default profile
CERTIFICATE (X509V3 – ES256)	
CLIENT IP/IP RANGE	
Client Authentication method	private_key_jwt (the client sends its credentials as a JWT)

APPENDIX C: SNIPPET TO GET AN ACCESS TOKEN FROM WIPO DEV OPENAM

The bash script below is an example of WIPO authentication request using the IPO private key:

```
#!/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}"
```

APPENDIX D: PUBLIC HAGUE PLATFORM API



Note: additional information will be added to later versions of this document.