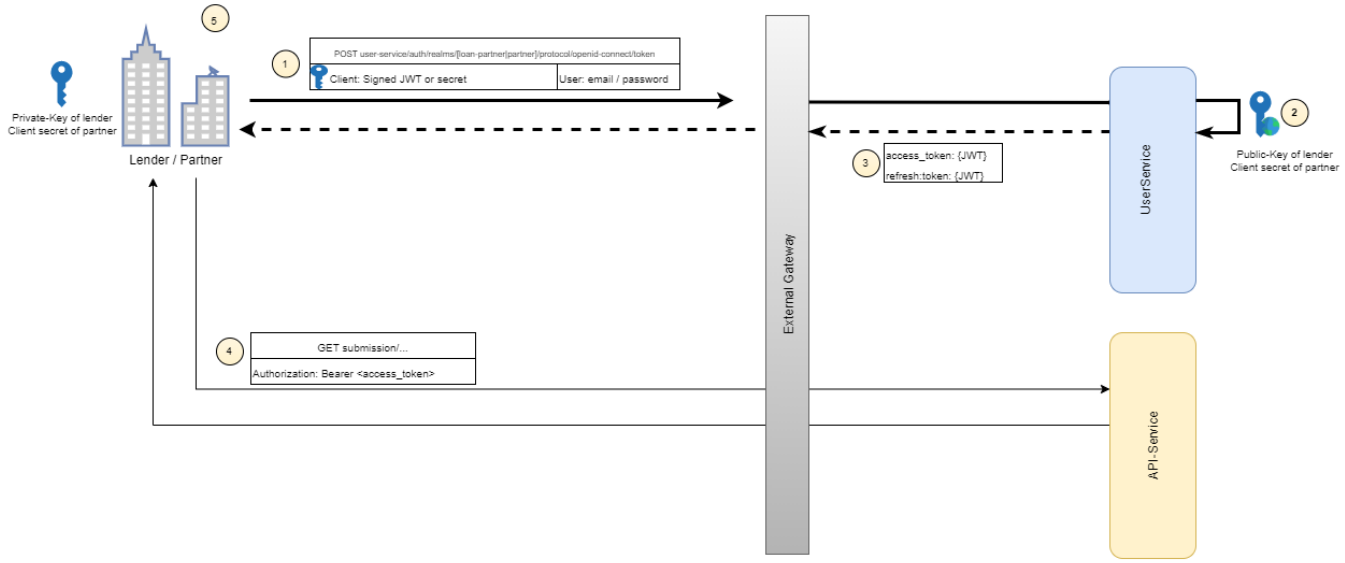
Authentication of Lenders and Partners using the api.interhyp.de (eng.)

- Context
  - Procedure Overview
- Client Setup for lenders
  - Key Generation (openssl)
  - Key Transmission and Client Creation
- Authentication (Java)
- Request Access Token (with curl)
  - Authentication Server Response
  - Time synchronization

Context

Procedure Overview



One-time Registration of Lender with Interhyp

- Generation of private and public key.
- Lender sends public key to Interhyp in order to update the lender client.
- Partner gets client secret from Interhyp
- Lender/partner defines email address for the technical user.
- Interhyp creates technical user in Keycloak.

Procedure

1. Lender/partner sends a query compliant with the OAuth2 standard to Interhyp.
   The query includes:
   - Lender: A JWT signed with the private key of the lender, the name of the lender and an expiration date.
   - Partner: The client secret assigned by Interhyp
   - Email address and password of the technical user.
2. Interhyp verifies the token including the expiration date (only for lenders), the name of the lender/partner, and the name and password of the technical user.
3. Interhyp returns a time bounded Access Token JWT / Refresh Token in case of successful validation.
4. The lender/partner uses the access token to access the API.
5. Access token have a time bounded validity. After the expiration date the lender/partner uses Access-grant_type=refresh_token, refresh_token=<x> to create a new access token.

⚠ This documentation depicts the authentication mechanism for the lender using the submission-API and the authentication of partners for the use of the interestcalculator-API. The steps mentioned in the document below always refer to both variants if not explicitly mentioned differently.

Client Setup for lenders

In order to receive a token via the Client Authentication API (https://tools.ietf.org/html/rfc7523), Interhyp must first be sent the public key. This section depicts a method of creating a public key. Furthermore it also describes how to transmit the key to Interhyp.

Terms

JWT

The abbreviation JWT stands for JSON Web Token. Further information can be found on jwt.io and RFC 7519.

Access Token

Access token are used in conjunction with Open ID Connect. You receive an access token once you logged in. To prove that you are logged in, the access token is send to the API with each interaction. Interhyp uses access token that are formatted as JWT.

Client Authentication JWT

For the login you have to send username and password to a server. However Interhyp requires more cryptographically secured data for the login that has to be formatted as JWT and issued by the operator of the API (Lender / Partner)

Private and Public Key

Asymmetric Cryptography uses two complementary keys. One key is private and the other is public. In principle the public key can be used to encrypt data that can only be decrypted by the private key. On the other hand side the private key can be used to sign data that can be verified by the public key.

Authentication and Registration of Lender

For the successful authentication, your technical system has to be registered as a client. The following steps describe the process in detail:

1. Lender generates private and public key.
2. Lender sends the public key to Interhyp. The public key will be deposited at the client.
3. Lender states an email for the technical user
4. Interhyp creates a technical user in Keycloak.

Key Generation (openssl)

The tool openssl is used to generate the public and private key.

```
openssl genpkey -algorithm RSA -out private_key.pem -pkeyopt rsa_keygen_bits:4096
```

Openssl creates a file `private_key.pem` that contains the public as well as the private key. The file must be stored in a secure location and may not be shared (even with Interhyp). Likewise, Interhyp can not help with the recovery of the file.

The file may look as follows:

private_key.pem

```
-----BEGIN PRIVATE KEY-----
MIIJQwIBADANBgkqhkiG9w0BAQEFAASCCS0wggkpAgEAAoICAQCs9eTSXVwxL/TZ
gA0ZCq49L2RS5ByEmW0LnFZ7juz7y8NhL0HX9wzUy13TpHwvVhMGQZKqsaV7AwKo
Jv+h7OsIbueFWSPRRW4Hc3wMwMVVRn323q8pZ3+zav025P6eW4qyV7iww6FcuCYW
07jFrw+jPSMPB1SuflRzeHxUvdh6MwGeM9c/2I08AiPUQiE+YD1aTHdHqfxS+Kjk
rPmvIzwEqRKM0LDkCl2zG98dS98UOt1DbE5mb3yq4cj0gBYUizYqSZkk3OBjSs9P
t8ogos9JNLBaS2NSmPtI9m+O2xRNBc9mhaHOpCJiBg9DioHxHTJFjQs8kcGM892F
AfZhk/DdqQ3bgiVeL9E8LMmK/9ZJpMXdTKPzJ9rpWzBFkplSBe/YqNqfTo79siM2
Jjg+eRHHMMe9pcfnfWZMKLCMFbIMr9TdfAQVqyCNjQSR6nBq/Tg6rFunS9hlFq55
WwjFQXi1dEghYoKLpYLe4IARkGgG2XZb1rsZpNBKtGa3IX5BdX5S1gWRl1NQ+2e9
W9OFGK4BoeMiAcffqcExK3iHEYnxVqnwBxmRy9nThQjoBg4w5kyA5QeUl7BD0wGl
pglFPqylVHYRAiuDoISgggVi3helebIxmxV9XIm1zxFVeS/wovDWfSaLwgK/c4+/
A2OZBzKdyKazi55VmPrnjce6rZIVRQIDAQABAoICAHLYI0P7ccfHTBFPkWLbUK6X
1Umk5bq2FU3sVRBd6sWWmvjg5Bt+kyqIjpiFU2ArCx7QU467vmFbslvs6Tifk8BZ
w6krczj1nQWdjPT62wywbduKKUyYyJZdqBzXGKoVppac4KmefHqQW9o0Yy//+aYu
ZiUIfRYQFAtIKWYrWJ5rV9uwGHDwXXdHBHtgHarJRHO86kbPfBervJl6sZ+1CZke
mo6d05bXuze+fft58XoBTAIE3FnwFRtyKHkQw5iIh0K089ciQmwrtFP+eVwKPxEg
xyhtflnN8qyT1adJmMU74mb0vGf1XE8yLthuE7IB7Ot5p8hkD9yD4Mb8cpiAHyI3
SVuKNF2zXRTW+LrntKrwVW3P/VooQMpXvCtfEpusaskKY81tDAsL6a+hVklsT4yF
ZXyhzpzCJt3zr9BzJxrmUaAcbtJ8BGquMPuL1b5P1AI1lOFoGBvkHTGL4IZVSU9X
5VzUWrfopdk1uQslHAu6oxWpnjGTxNECkZKxN4EaoFsQlAPYfsE/MgMq4wigpy3R
eR9r+iiyXC0UBZM0mX7eM9NwuP159lGVFyAcsBNNgnM+VtocQf9L4oiOeZaCqW5E
mngVC9io+iC04MNtUR5eYLLP/471YZ2pHFHKLZY3FWATsL2Js+rmPAbzBpp8Gqfg
FxfNKV8UWBCeuT7slcPRAoIBAQDkXkTb12hvYAubm3pib4GXX95q2n8RpRK2XIQH
dry0zHZzyV6jRFGAB9LXoIgMKxuCz3D0axZGSOyQXeHO/xoo82rjEaPJNUQ6G1eH
YVS/p3bPpWT5pU4z2nXmN4sbqlC0fB6rsku+XCYqKzKaXVqZEgl+MbnbsM64JZY1
xC1gk0rEHA5z8LaXFsnb2BhEOHBJNfTzQ6O3gQB8w8qZaQM1QvTkr7Up8PaCndw6
ruhKxD7s3CUYUiV9xTElmD0x/7YiTR1A+2omNUkbTCRecHABZwGLtJDc3tdB1dCI
7DjgIdR0T7iEnlvLHrU/GcMUe2R8ifvL7FzRLMRsQiVzddmfAoIBAQDB411mXqr+
VlzOYFdBCuYMmJ96BIFNU9NwwrNt055h4NPnqsYQihb1cDLKZ4z+mlRNICx1bp2V
ICZH1aQd8efh2cMWPEUPyeEWUbi5IT2ZfVzzAG/BPHR+S38/5/gbgGH3I5dCQ6Wc
MV7S+s98+fvp5ZQu3PCETXXe6zseWpjholsniwfOZbj4r+kkCARyGTEZMctw9p3/
0sWKwfcq8OHBMHjiSZZNgMktAZILDCuPbT0wF4ZyMHhIeKVxBAcDPV2eHlXcUfmW
b/I+sBClmzuvaP7H5jZ5+C0bijEauBpIopXUgAiSTxz64QSYaps4jPwZXFMCNknW
MCP8NpnsK26bAoIBAQCdZP+OiZNYF9lIQUuDZpjprNCN0tP7ZEtWVsNEpah/79/A
A8zvOBn35OyDYwgBYwCOEs0mGFx8yqwJADuuwBHWoZZMtA6vkb/ZUJjuHL4dbZ0s
ljDXAwC4KlxR9Hwy8Bq5mkZFThh4Neiz7Zvt2mOj1lOmnz0a1mxTbMxveCtSQZIp
QbEQZZpi3fj4RaLL+h4hx0dxFnBS/F1KoGMVgtQMNH620wdIxeQQuB5eb8h4J3O7
KiuC7LMcUPmDkL60pCpyz7VW4TES7mzZnIwAbvwzazwcfrHNGOVBYc1dYXn9uDNG
it+Av+hCQYLn36idiEIEt9FMhjaekH08pAZN4q6/AoIBACbiCP4ZqCMfgLcmnPj/
TYljJkBbHO88C2BGu6Mf2Fh6tSR1nAj/GIGaQaVTxPPeUV+yPxhB2bppWyP21VXo
CH/DoxsXdSYob0qBkxEVeHwKpxRYlxDipd4aF+fiBmDFi1/7tD7mwBSl1gVrpSbE
H2AV2XTYfCecPTI6bH2234gIfk1c1XDv93twE8pVDUcwhCcwVENQQflm9JTSN1ZW
Pn/2z5dZ+JmMZUvwT7CFfUrjNm0pmp3V2Jl4EH2suBdPPDIUlVV83xfrVWJ6/qun
iP2xib7fDTfCrJEkUpqwJx0/9dpK3u0wjnhFnh1O69gYdgpD+KWt63rX5q+S56V0
RFcCggEBAIZJkQ9vWT15a/rp5AJpzpG69Aph6m2h2kNjhVpx/kes9RkTA/8mSlLh
9td4uVcWaHx3nhEUgmTMc+9pNv4FFlPqjAIstvJOLJtHOYOuGPFTHGqIqF4+1r8P
4+tVxiMVS7xtKRC7EITiKoIcqDKvcAjFd3IsJ0qE37+MX1O3dGfgLLbjatFXgJEp
UfTWHaJq9aldF1TNh7nbyh8mNCEWKnReoMjnzK21XjYOid/1IYyseR6inMIoUjdv
61DhC1J60pkDkAaydd8KCjqZOIljDJ1u6/WNK3PEFhlXaCSWkjJTAi4c4qfVK1bE
ct65CeWXGqil/BPsvlbGM1lT61b9CHM=
-----END PRIVATE KEY-----
```

Key Transmission and Client Creation

The public key must be transmitted to Interhyp. First, the public key has to be extracted from the file mentioned above. A fingerprint of the public key is also created to validate the correct transmission.

```
openssl rsa -pubout -in private_key.pem -out public_key.pem
openssl dgst -sha256 public_key.pem > public_key_digest.txt
```

The file may look as follows:

public_key.pem

```
-----BEGIN PUBLIC KEY-----
MIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEArPXk0l1cMS/02YANGQqu
PS9kUuQchJltC5xWe47s+8vDYS9B1/cM1Mtd06R8L1YTBkGSqrGlewMCqCb/oezr
CG7nhVkj0UVuB3N8DMDFVUZ99t6vKWd/s2r9NuT+nluKsle4sMOhXLgmFtO4xa8P
oz0jDwdUrn5Uc3h8VL3YejMBnjPXP9iNPAIj1EIhPmA9Wkx3R6n8Uvio5Kz5ryM8
BKkSjNCw5ApdsxvfHUvfFDrdQ2xOZm98quHI9IAWFIs2KkmZJNzgY0rPT7fKIKLP
STSwWktjUpj7SPZvjtsUTQXPZoWhzqQiYgYPQ4qB8R0yRY0LPJHBjPPdhQH2YZPw
3akN24IlXi/RPCzJiv/WSaTF3Uyj8yfa6VswRZKZUgXv2Kjan06O/bIjNiY4PnkR
xzDHvaXH531mTCiwjBWyDK/U3XwEFasgjY0Ekepwav04Oqxbp0vYZRaueVsIxUF4
tXRIIWKCi6WC3uCAEZBoBtl2W9a7GaTQSrRmtyF+QXV+UtYFkZdTUPtnvVvThRiu
AaHjIgHH36nBMSt4hxGJ8Vap8AcZkcvZ04UI6AYOMOZMgOUHlJewQ9MBpaYJRT6s
pVR2EQIrg6CEoIIFYt4XpXmyMZsVfVyJtc8RVXkv8KLw1n0mi8ICv3OPvwNjmQcy
ncims4ueVZj6543Huq2SFUUCAwEAAQ==
-----END PUBLIC KEY-----
```

public_key_digest.txt

```
SHA256(public_key.pem)= b063a92813a9799743781356bfa986db490522107cfd9cf4aae58717878d7027
```

The two generated files `public_key.pem` and `public_key_digest.txt` can now be sent as an email attachment to your contact at Interhyp. If you are creating an account for the production system, you must provide an email address that you can access. You will then be prompted to open the file `public_key_digest.txt` and read it on the phone. This is important to verify that the content has been transferred correctly.

The successful public key transmission will be followed by the generation of the client. Interhyp will send you the denotation of the client. The client denotation should then be used in the following code examples instead of `"example-bank"` or `"example-partner"`.

Authentication (Java)

To request a Token you first have to enter the private key to create a JWT. The private key is used to sign the JWT. Only a signed JWT can request an access token.

The access token can be used to access the API until the expiration of the token. Afterwards a new token has to be requested and signed with a new JWT. A token can only be used once. This is ensured by the token-ID, which must be different in each JWT.

The following code examples describe the process in java:

Read Private Key

```
String privateKeyPem = new String(Files.readAllBytes(Paths.get("private_key.pem")), StandardCharsets.
US_ASCII);
byte[] privateKeyDer = Base64.getMimeDecoder().decode(privateKeyPem.replaceAll("-----(BEGIN|END) PRIVATE
KEY-----\n?", ""));
PrivateKey privateKey = KeyFactory.getInstance("RSA").generatePrivate(new PKCS8EncodedKeySpec(privateKeyDer));
```

Create Client Authentication JWT

The code below uses the private key (`privateKey`) to create the client authentication JWT. For this you can use the class `Jwts` from the Java JWT library.

Creation ClientAuthJWT for lenders

```
String clientAuthenticationJwt = io.jsonwebtoken.Jwts.builder()
        .setSubject("example-bank")
    .setIssuer("example-bank")
    .setId(UUID.randomUUID().toString())
    .setAudience("https://api-test.interhyp.de/user-service/auth/realms/loan-partner/protocol/openid-connect
/token")
    .setExpiration(new Date(Integer.MAX_VALUE * 1000L)) // this is roughly year 2038, in prodcution, please
use a short expiration time, i.e. Date.from(Instant.now().plusSeconds(30))
    .signWith(SignatureAlgorithm.RS256, privateKey)
    .compact();
```

The example token `clientAuthenticationJwt` for the lender case looks as follows (The private key and the ID from above is used):

Example token for lenders

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.
eyJzdWIiOiJleGFtcGxlLWJhbmsiLCJqdGkiOiJ1bmlxdWUtc3RyaW5nIiwiYXVkIjoiaHR0cHM6Ly9hcGktdGVzdC5pbnRlcmh5cC5kZS91c2
VyLXNlcnZpY2UvYXV0aC9yZWFsbXMvbG9hbi1wYXJ0bmVyL3Byb3RvY29sL29wZW5pZC1jb25uZWN0L3Rva2VuIiwiZXhwIjoyMTQ3NDgzNjQ3
fQ.iZRo0toZklO0VY_isoyb1oSO3z1SjqpqdrrCW3Ojr4H5iln26dL47giQGXaaMFpX5e6PN1PVml4bp_wplcjR-
PO1_YiMcFmcI883VUVSGFJDpbJ3QdRgNH99Ano-
WZsHOZ7XBCFIFkBCdTOqAJvoGm1uYVjp8WTSGebhMu_KeD0ONCiUiwTKM0UmdW5vF65uziWQ1wP7z0mPpdmuk-
ckCxkFVtLTWqmKUp5xwhpKLvFI5I0NabM1VlgnkjM0V0IXMy-6YV5HjI0-
EHOzohDT_ghbsuT_iBOBpDHXaxa0LMUF65qqrxfmR7ORVZhItpURPPwhoujpU4xvSpLFP1GYaM8RZCkIBfNGF3k5jkQ9kC1rUKBhX2WTohar4y
b5lofgJCAVQ4aEOL9FVJJXxaen6sJYtQC6wOhuWt9_xQiEEfc5c6qSGzd7vVR077nW0_qQ-ij7qsns8avonaTxTGQnn_uXHlEj-
w7LTH9CFaDr27eli6_gNc-heAVK6PQQKye6xmy0bynwG_DPjUVtPrl2BEzfzMgkta5duwLPn29XZ7NaK2tqQeQHwEk7cYxDt-
n9Ta0plKGJYIq4gtFsy5pI8R4W2IXmzEo9UjhiWBIvTzQMlyyaDzUmBApt5k-LR6WbFWiE2JN8r2gQ11pQFUnMc8o1E0pY9kzeyRwIAXMZfRg
```

The website jwt.io can be used to view the content of JWTs. The JWT is inserted in the window "Encoded".

The client authentication JWT from above corresponds to the following JSON header and payload:

Header

```
{
        "alg": "RS256"
}
```

Payload of a lender token

```
{
  "sub": "example-bank",
  "iss": "example-bank",
  "jti": "unique-string",
  "aud": "https://api-test.interhyp.de/user-service/auth/realms/loan-partner/protocol/openid-connect/token",
  "exp": 2147483647
}
```

The value "exp" describes the expiration of the token in unix time (Unix converter: unixtime.de).

Request Access Token (with curl)

The example Client Authentication JWT created above can be used to request an Access Token JWT corresponding to RFC 7523 for the user `exampleuser@bank.example.com` and password `SecUre!P4ssw0rd` (Translates to `SecUre%21P4ssw0rd` using the URL-Encoding).

Lender:

Since Client Authentication is only valid once, the following examples can no be used directly. The value of the `client_assertion` parameters must be replaced with a newly created Client Authentication JWT.

Requesting an access token with cURL for lenders

```
curl --request POST \
     --url https://api-test.interhyp.de/user-service/auth/realms/loan-partner/protocol/openid-connect/token \
     --data "grant_type=password&username=exampleuser%40bank.example.com&password=SecUre%
21P4ssw0rd&client_assertion_type=urn%3Aietf%3Aparams%3Aoauth%3Aclient-assertion-type%3Ajwt-
bearer&client_assertion=eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.
eyJzdWIiOiJleGFtcGxlLWJhbmsiLCJqdGkiOiJ1bmlxdWUtc3RyaW5nIiwiYXVkIjoiaHR0cHM6Ly9hcGktdGVzdC5pbnRlcmh5cC5kZS91c2
VyLXNlcnZpY2UvYXV0aC9yZWFsbXMvbG9hbi1wYXJ0bmVyL3Byb3RvY29sL29wZW5pZC1jb25uZWN0L3Rva2VuIiwiZXhwIjoyMTQ3NDgzNjQ3
fQ.iZRo0toZklO0VY_isoyb1oSO3z1SjqpqdrrCW3Ojr4H5iln26dL47giQGXaaMFpX5e6PN1PVml4bp_wplcjR-
PO1_YiMcFmcI883VUVSGFJDpbJ3QdRgNH99Ano-
WZsHOZ7XBCFIFkBCdTOqAJvoGm1uYVjp8WTSGebhMu_KeD0ONCiUiwTKM0UmdW5vF65uziWQ1wP7z0mPpdmuk-
ckCxkFVtLTWqmKUp5xwhpKLvFI5I0NabM1VlgnkjM0V0IXMy-6YV5HjI0-
EHOzohDT_ghbsuT_iBOBpDHXaxa0LMUF65qqrxfmR7ORVZhItpURPPwhoujpU4xvSpLFP1GYaM8RZCkIBfNGF3k5jkQ9kC1rUKBhX2WTohar4y
b5lofgJCAVQ4aEOL9FVJJXxaen6sJYtQC6wOhuWt9_xQiEEfc5c6qSGzd7vVR077nW0_qQ-ij7qsns8avonaTxTGQnn_uXHlEj-
w7LTH9CFaDr27eli6_gNc-heAVK6PQQKye6xmy0bynwG_DPjUVtPrl2BEzfzMgkta5duwLPn29XZ7NaK2tqQeQHwEk7cYxDt-
n9Ta0plKGJYIq4gtFsy5pI8R4W2IXmzEo9UjhiWBIvTzQMlyyaDzUmBApt5k-LR6WbFWiE2JN8r2gQ11pQFUnMc8o1E0pY9kzeyRwIAXMZfRg"
```

Partner:

Requesting an access token with cURL for partners

```
curl --request POST \
     --url https://api-test.interhyp.de/user-service/auth/realms/partner/protocol/openid-connect/token \
     --data "grant_type=password&client_id=example-partner&client_secret=example-partner-
dummysecret&username=exampleuser%40partner.example.com&password=SecUre%21P4ssw0rd"
```

Authentication Server Response

The response includes the current access and refresh token.

```
{
        "access_token":"
eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJpMFZVcVg5VndnUVRJN1oxZm43UHY0Vl9xUWdwSjJVWHM4MXd3b1dtNDNnIn
0.
eyJqdGkiOiI2NDYzZTA2NC05MGM5LTQzNmQtOGUxYy1hMmQ4NGI1NTFmZGEiLCJleHAiOjE1MjQ0ODM0OTAsIm5iZiI6MCwiaWF0IjoxNTI0ND
gzMTkwLCJpc3MiOiJodHRwczovL2FwaS10ZXN0LmludGVyaHlwLmRlL3VzZXItc2VydmljZS9hdXRoL3JlYWxtcy9sb2FuLXBhcnRuZXIiLCJh
dWQiOiJleGFtcGxlLWJhbmsiLCJzdWIiOiJkMjVhNzg5My0yN2RjLTQ1NTgtYmRmZC1jYTE3NmVlZDdlMTUiLCJ0eXAiOiJCZWFyZXIiLCJhen
AiOiJleGFtcGxlLWJhbmsiLCJhdXRoX3RpbWUiOjAsInNlc3Npb25fc3RhdGUiOiIwYzk2ZGE1OS0yMGM3LTQ0OTAtOWI0Ny0yNjhkOTg5Mzgz
ZDUiLCJhY3IiOiIxIiwiYWxsb3dlZC1vcmlnaW5zIjpbXSwicmVhbG1fYWNjZXNzIjp7InJvbGVzIjpbInVtYV9hdXRob3JpemF0aW9uIl19LC
JyZXNvdXJjZV9hY2Nlc3MiOnsiYWNjb3VudCI6eyJyb2xlcyI6WyJtYW5hZ2UtYWNjb3VudCIsIm1hbmFnZS1hY2NvdW50LWxpbmtzIiwidmll
dy1wcm9maWxlIl19fSwiYXVkIjoiYWxsIiwicHJlZmVycmVkX3VzZXJuYW1lIjoiZXhhbXBsZXVzZXJAYmFuay5leGFtcGxlLmNvbSIsImVtYW
lsIjoiZXhhbXBsZXVzZXJAYmFuay5leGFtcGxlLmNvbSJ9.G3FjnOCkiQ0rt4gAd2T9pZeZk56hKr7qYJdrVHE-OLs39-
6Mu602Ygqdpwp2wb3cPFXUFkgliVgP1a6B_rsxUIe2NTexP2Mn4FRgXRUmeGK1XGxMd2BArCXvCk5EDB7gFEWtlF2609GXiWNvQP5KW2LLVTqQ
eTCIGY-EBTGzgP-0aHj-GGPeowOcESyIeSQ3vMleE-
_JAKX7VhAIoGkmibPFdVcpkxeWypcuzHNQ415_Pv3kli57PX9yzUnJJdco5Tj6oslEYbHb4Kxi0H8WSwI-
lWZZ9JMmPP7Rya6UQkqtreSQ_R8FildhoOOjyC1ydG67Pr4f_FfiozohWugeiA",
        "expires_in":300,
        "refresh_expires_in":900,
        "refresh_token":"
eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJpMFZVcVg5VndnUVRJN1oxZm43UHY0Vl9xUWdwSjJVWHM4MXd3b1dtNDNnIn
0.
eyJqdGkiOiIxMDBlNWFmNS1lN2Q2LTQ2MTYtODQ1Yi01Mjc3YTc0ZmJiMDkiLCJleHAiOjE1MjQ0ODQwOTAsIm5iZiI6MCwiaWF0IjoxNTI0ND
gzMTkwLCJpc3MiOiJodHRwczovL2FwaS10ZXN0LmludGVyaHlwLmRlL3VzZXItc2VydmljZS9hdXRoL3JlYWxtcy9sb2FuLXBhcnRuZXIiLCJh
dWQiOiJleGFtcGxlLWJhbmsiLCJzdWIiOiJkMjVhNzg5My0yN2RjLTQ1NTgtYmRmZC1jYTE3NmVlZDdlMTUiLCJ0eXAiOiJSZWZyZXNoIiwiYX
pwIjoiZXhhbXBsZS1iYW5rIiwiYXV0aF90aW1lIjowLCJzZXNzaW9uX3N0YXRlIjoiMGM5NmRhNTktMjBjNy00NDkwLTliNDctMjY4ZDk4OTM4
M2Q1IiwicmVhbG1fYWNjZXNzIjp7InJvbGVzIjpbInVtYV9hdXRob3JpemF0aW9uIl19LCJyZXNvdXJjZV9hY2Nlc3MiOnsiYWNjb3VudCI6ey
Jyb2xlcyI6WyJtYW5hZ2UtYWNjb3VudCIsIm1hbmFnZS1hY2NvdW50LWxpbmtzIiwidmlldy1wcm9maWxlIl19fX0.
TaxAGcdPsJQ31gnmHVUCfUU5TZp8cPtvnJsk9v0ny3Jt1SvXpNCceiwrqbsLCKArl5thr5ZYsP9vYFfBjKFaKvmCWw97WzCpdv0VxJAbtNILQl
jKxAxBiQLY97EaQlq9JKqooMtZcFDOdxel7WugVYAcY3PILrtQnmsYP_CnpuGg4tHyvr0LaFeIWcfTYMYp07lkdLVhZwEt9lX3TgoFDtOJaNU2
VUBxTRDedn81hk71kdrxHZH8jnyarSGJrmJOz8Y9CPKgAgO15CCIa-
1CpygiyCZr0f09uDb09Lg7c8tWgCq4YzEK8D4k4txp5QWX3CiHmMYhFzKAKuEFpYEw",
        "token_type":"bearer",
        "not-before-policy":0,
        "session_state":"0c96da59-20c7-4490-9b47-268d989383d5"
}
```

Time synchronization

Both token are time bounded (The signed JWT to request the Access Token as well as the Access token itself). This procedure works only if the servers involved agree on when these validity periods are. It is mandatory that the server clock is kept up-to-date.