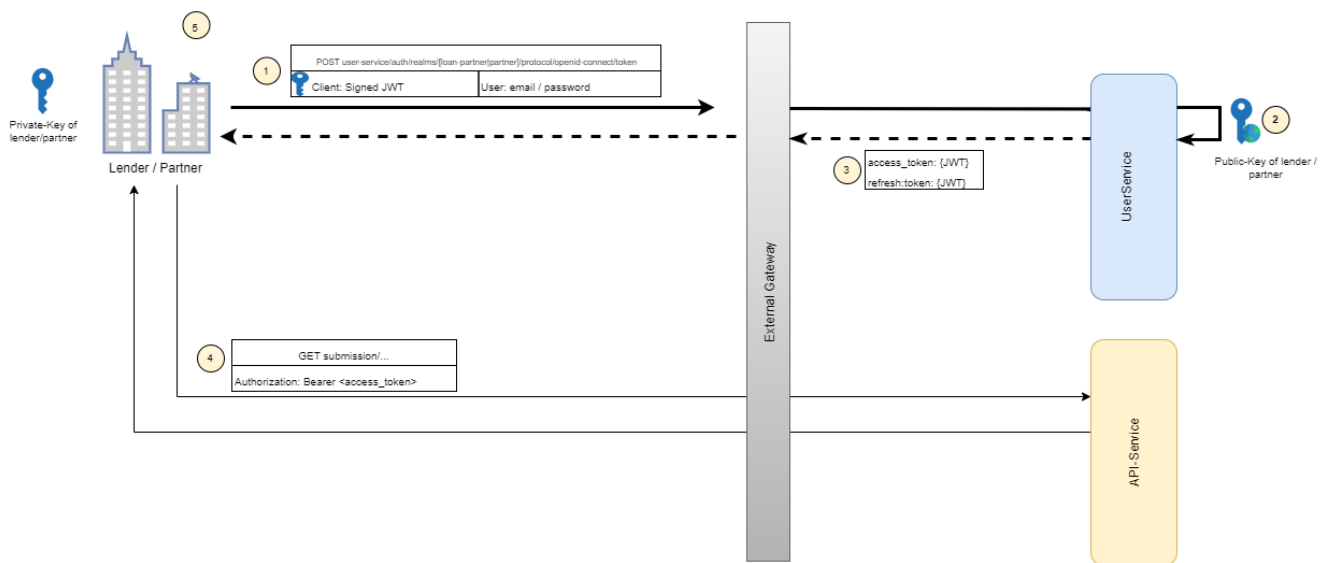


# Authentication of Lenders and Partners using the api. interhyp.de (eng.)

- Context
  - Procedure Overview
- Client Setup
  - 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.
- Lender 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:
  - A JWT signed with the private key of the lender/partner, the name of the lender/partner and an expiration date.
  - email address and password of the technical user.
2. Interhyp verifies the token, the name of the lender/partner, the expiration date, 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=<> 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

In order to receive a token via the Client Authentication API (<https://tools.ietf.org/html/rfc7523>), Interhyp must first be send 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](https://jwt.io) and [RFC 7519](https://tools.ietf.org/html/rfc7519).

### 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/Partner

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

1. Lender/Partner generates private and public key.
2. Lender/Partner sends the public key to Interhyp. The public key will be deposited at the client.
3. Lender/Partner 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-----
MIIEJQwIBADANBgkqhkiG9w0BAQEFAASCCS0wggkpAgEAAoICAQCs9eTSXVwxL/TZ
gAOZCq49L2RS5ByEmW0LnFZ77juz7y8NhL0HX9wzUy13TpHwvVhMQGZKqsaV7AwKo
Jv+h70sIbueFWSPPRRW4Hc3WmMVVRn323q8pZ3+zav025P6eW4qyV7iww6FcuCYW
07jFrw+jPSMPB1Suf1RzeHxUvdh6MwGeM9c/2I08AiPUQiE+YD1aTHdHqfxS+Kjk
rPmvIzwEqRKM0LDkC12zG98dS98UOt1DbE5mb3yq4cj0gBYUizYqSZkk30BjSs9P
t8ogos9JNLBaS2NSmPtI9m+02xRNBc9mhaHOpCJiBg9DioHxHTJfJqS8kcGM892F
AfZhk/DdQq3bgiVeL9E8LMmK/9ZJpMXdTKPzJ9rpWzBFkplSBe/YqNqfTo79siM2
Jjg+eHHHMe9pccfnfWZMKLCMFbIMr9TdfAQVqyCNjQSR6nBq/Tg6rFunS9hlFq55
WwjfQXil1dEghYoKlpYle4IARkGgG2XZb1rsZpNBktGa3IX5BdX5S1gWRl1NQ+2e9
W9OFGK4BoeMiAcffqccEkX3iHEYnxVqnWbXmRy9nThQjjoBg4w5kyA5QeU17BD0wG1
pglFPqylVHYRAiuDoISgggVi3helebIxmxV9XIm1zxFVeS/wovDWfSaLwgK/c4+/
A2OZBzKdyKazi55VmPrnjce6rZIVRQIDAQABAOICAHL1YI0P7ccfHTBFPKWLBUK6X
1UmK5bq2FU3sVRBd6sWmwjg5Bt+kyqIjpiFU2ArCx7QU467vmFbslvs6Tifk8BZ
w6krczjlnQWdjPT62wywbduKKUyYyJZdqBzXGKoVppac4KmfHqQW9o0Yy//+aYu
ZiUIfRQFAtIKWYrWJ5rV9uwGHDWXXdHBHtgHarJRHO86kbPfbervJl6sZ+1CZke
mo6d05bXuZe+fft58XoBTAIE3FnwFRtyKHkQw5iIh0K089ciQmwrtFP+eVwKPxeg
xyhtflnN8qyTladJmMU74mb0vGf1XE8yLthuE7IB7Ot5p8hkD9yD4Mb8cpiAHyI3
SVuKNF2zXRTW+LrntKrwVW3P/VooQMpxVctfEpusaskKY81tDAsL6a+hVklst4yF
ZxyhczpZCJt3zr9BzJxrmUaAcbtJ8BGquMPuL1b5P1AI1lOfOGbvKHTGL4IZVSU9X
5VzUWrfopdk1uQ1LHAu6oxWpnjGTxNECkZKxN4EaoFsQlAPYfsE/MgMq4wigpy3R
er9r+iiyXC0UBZM0mX7eM9NwuP159lGVfyAcsBNNngnM+VtocQf9L4oiOeZaCqW5E
mngVC9io+iC04MNtUR5eYLLP/471YZ2pHFHKLZY3FWATSL2Js+rmPAbzBpp8Gqfg
FxfNKV8UWBCEut7slcPRAoIBAQDkXkTb12hvYAubm3pib4GXX95q2n8RpRK2XIQH
dry0zhZzyV6jRFGAB9LXoIgMKxuCz3D0axZGS0yQXeHO/xoo82rjEaPjNUQ6G1eH
YVS/p3bPpWT5pU4z2nXmN4sbqlC0fB6rsku+XCyqKzKaXVqZEgl+MbnbsM64fZY1
xC1gk0rEHA5z8LaXfSnb2BhEOHBjNfTzQ603gQB8w8qZaQM1QvTkr7Up8PaCndw6
ruhKxd7s3CUYUiv9xTElmd0x/7YiTR1A+2omNUkbTCRecHABZwGLtJdC3tdBldCI
7DjgIdR0T7iEnlVlHrU/GcMUe2R8ifvL7FzRLMRsQiVzddmfAoIBAQDB41lmXqr+
VlzOYfBDCuYmmJ96BIFNU9NwrrNt055h4NPngsYQihblcDLKZ4z+mlRNICx1bp2V
ICZHlaQd8efh2cMWPEUPyEwUbi5IT2zfVzAG/BPHR+S38/5/gbgGH3I5dCQ6Wc
MV7S+s98+fvp5ZQu3PCETXXe6zseWpjholnsniwfOZbj4r+kkCARYGTEZMctw9p3/
0sWKwfcq8OHBMHjiSZZNgMktAZILDCuPbT0wF4ZyMHhIeKVxBACDPV2eHlXcUfmW
b/I+sBCLmzuvaP7H5jZ5+C0biJEauBpIopXUGAiSTxz64QSYaps4jPwZXFMcnknW
MCP8NpnsK26bAoIBAQCdZP+OiZNYF9lIUuUZpjpRNCN0tP7ZEtWVsNEpah/79/A
A8zvOBn35OyDYwgBYwCOEs0mGFx8yqwJADuuwBHWoZZMtA6vkb/ZUJjuHL4dbZ0s
ljDXAwC4KlxR9Hwy8Bg5mkZFThh4Neiz7Zvt2m0j1l0mnz0a1mxTbMxveCtSQZIp
QbEQZ2Zpi3fj4RaLL+h4hx0dxFnBS/F1koGMVgtQMNH620wdIxeQQUB5eb8h4J3O7
KiuC7LMcUPmDkL60pCpyz7VW4TES7mzZnIwAbvwazwzfrHNGOVBYclYXn9uDNG
it+Av+hCQYLn36idiEIEt9FMhjaekH08pAZN4q6/AoIBACbiCP4ZqCMfgLcmnPj/
TY1jJkBBhO88C2BGU6mf2Fh6tSRlnAj/GIGaQaVTxPpEUV+yPxb2bpbWyp21VXo
CH/DoxsXsYob0qBkxEVeHwKpxRylxDipd4aF+fiBmDFil/7td7mwBS1lgVrpSbe
H2AV2XTYfCecPTI6bH2234gIfk1clXDv93twE8pVDUcwhCcwVENQQf1m9JTSN1ZW
Pn/2z5dZ+JmMZUvwt7CFfUrjNm0pmp3V2Jl4EH2suBdPPDIU1VV83xfrVWJ6/qun
iP2xib7fDTfCrJEkUpqWjx0/9dpK3u0wjnhFnhl069gydgpD+Kwt63rX5q+S56V0
RfCcgEBAIZJkQ9vWT15a/rp5AJpZpG69Aph6m2h2kNjhVpx/kes9RkTA/8mS1Lh
9td4uVcWaHx3nhEUGmTmc+9pNv4FFlPqjAIstvjOLJtHOYOuGPFTHGqIqF4+1r8P
4+tVxiMVS7xtKRC7EITiKoIcqdKvcA jFd3IsJ0qE37+MX103dGfgLLbjatFXgJEp
UfTWHaJq9aldF1TNh7nbYh8mNCEWKnReoMjnzK21XjYoid/1IYySeR6inMioUjdV
61DhC1J60pkDkAaydd8KcjqZO1ljDjlu6/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-----
MIICIjANBgkqhkiG9w0BAQEFAAACg8AMIICGKCAgEArPXk011cMS/02YANGQqu
PS9kUuQchJltC5xWe47s+8vDYS9B1/cM1Mtd06R8L1YTBkGSqrGlewMCqCb/oezr
CG7nhVkj0UVuB3N8DMDfVUZ99t6vKWd/s2r9NuT+nlUKsle4sMOhXLgmFtO4xa8P
oz0jDwdUrn5Uc3h8VL3YejMBnjPXP9iNPAIj1EIhPmA9Wkx3R6n8Uvio5Kz5ryM8
BKkSjNCw5ApdsxvfHUVfVFDrdQ2xOZm98quHI9IAWFI52KkmZJNzgY0rPT7fKIKLP
STSwWktjUpj7SPZvjtsUTQXPZoWhzqQiYgYPQ4qB8R0yRY0LPJHBjPPdhQH2YZPw
3akN24I1Xi/RPCzJiv/WSaTF3Uyj8yfa6VswRZKZUGXv2Kjan06O/bIjNiY4PnkR
xzDHvaXH53lmTCiwjBWyDK/U3XwEFasgjY0Ekepwav04Oqxbp0vYZRaueVsIxUF4
tXRIIWKC16WC3uCAEZBoBtl2W9a7GaTQSRmtYF+QXV+UtYFkZdTUptnvVvThRiu
AahjIgHH36nBMst4hxGJ8Vap8AcZkcvZ04UI6AYOMZMgOUHLJewQ9MBpaYJRT6s
pVR2EQIrg6CEoIIFYt4XpXmyMZsVfVYJtc8RVXkv8KLwln0mi8ICv3OPvwNjmQcy
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")
    .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 prodcuton, please
use a short expiration time, i.e. Date.from(Instant.now().plusSeconds(30))
    .signWith(SignatureAlgorithm.RS256, privateKey)
    .compact();
```

## Creation ClientAuthJWT for Partner

```
String clientAuthenticationJwt = io.jsonwebtoken.Jwts.builder()
    .setSubject("example-partner")
    .setId(UUID.randomUUID().toString())
    .setAudience("https://api-test.interhyp.de/user-service/auth/realms/partner/protocol/openid-connect/token")
    .setExpiration(new Date(Integer.MAX_VALUE * 1000L)) // this is roughly year 2038, in prodction, 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.eyJzdWIiOiJleGFtcGxlLWJhbmsiLCJqdGkiOiJlbmlxdWUtc3RyaW5nIiwiaXVkiOiJoiaHR0cHM6Ly9hcGktZGVzdC5pbmRlcmh5cC5kZS91c2VyLXNlcnZpY2UvYXV0aC9yZWZsbXNlbnR5bWVjZC90b2t1biIsImV4cCI6MjE0NzQ4MzY0N30.fQ.iZRo0toZkl00VY_isoyb1oS03z1SjqpgdrrCW3Oj4H5iln26dL47giQGxAMFpX5e6PN1PVm14bp_wplcjr-PO1_YiMcFmcI883VUVSGFJDpbJ3QdRgNH99Ano-WZsHOZ7XBCFIKBCdToqAJvoGmluYVjp8WTSgebhMu_Ked0ONCiuIwTKM0UmdW5vF65uziWQ1wP7z0mPpdmuk-ckCxkFVtLTWgmKUP5xwhpKLVFI5I0NabM1VlgnkjM0V0IXMy-6YV5HjI0-EHOzohDT_ghbsuT_iBOBPdHXaxa0LMUF65qqrxfmR7ORVZhItPURPPwhoujpU4xvSpLFP1GYaM8RZCkIBfNGF3k5jkQ9kClrUKBhX2WTohar4yb5lofgJCAVQ4aEOL9FVJJXxaen6sJYtQC6wOhuWt9_xQiEEfc5c6qSGzd7vVR077nW0_qQ-ij7qsns8avonaTxTGQnn_uXH1Ej-w7LTH9CFADr27eli6_gNc-heAVK6PQQKye6xmy0bynwG_DPjUVtPr12BEzfzMGkta5duwLPn29XZ7NaK2tqQeQHwEk7cYxDt-n9Ta0p1KGJYIq4gtFsy5pI8R4W2IXmzEo9UjhiWBIVtZQMlyyaDzUmBapt5k-LR6WbFWiE2JN8r2gQ11pQFUnMc8o1E0pY9kzeyRwIAXMZfRg
```

For the partner the token looks as follows:

## Example token for partner

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJleGFtcGxlLXBhcnR5bWVjZC90b2t1biIsImV4cCI6MjE0NzQ4MzY0N30.JSfAJYoLniVbie2fEZaxDisKq7H04ZdZQRvOz3Z1ST98aXTiXWUwbHzgk69G0TuJPZk2M95400TMOqBAsiTupiFkXJzGtG-fKMj-gT90_ScAoYU3JQrPgkzAqmskhbV1vDdW0oZHmWGYh_u0WUUbShEe3JTCthwZlOTZUGIcFC_NteHoYCEzg3gyHqAqhsy91PEVWxIY8-W2sJ_M_a2mDwMTkhXA708YDLvsQ5DK4Dxx41KCzm4mxltIHSsRuSrg9JQ31NrDFKntmFwCYlGhMlHtwUGiYPY9XTAH1Z_woZvYV7K7ypOjosi6Vadhxt8Wg-unAOcJa03GKmvH4cQ-cL2RAiggR8AYRLIVM9soxQjYK8C3TA2S0Pl6tfDgw1WolEf2d9TdXydXcTn2zz0ADLJ21uLo3MI-o5Q36ZnBkzVoKQv1sJSZLdtzLdON8iWFE1GZb8SKKpsx6UZPNzNX09CwYxzkKKd7WfVXFCMAhOJMSsq9q5d_vd7xdyhGNhaOReUrxDLcfABz4GnvxrXRTehMatm-HLo21sVdYAAkM7B-gLiEdRPOQgUnLERTi1X6vlpbdcRg8tLy7yOKTXdG1vUaAnIXIogQDLnL5dzWK3IzHlnmU9kcDT9G1tgBpsrKpwdwAHZ2iM_2_VYLZjslR3IVPEI6Yr45zV9bWGpnK2w
```



The website [jwt.io](https://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",
  "jti": "unique-string",
  "aud": "https://api-test.interhyp.de/user-service/auth/realms/loan-partner/protocol/openid-connect/token",
  "exp": 2147483647
}
```



