OAuth 2.0 API

Version 4.0.1 | Status Live | Published date 9/4/2025

Intro

This guide is covering how to authenticate your application and retrieve an application access token or customer access token before you access ING's APIs. Based on the type of API you want to access, please refer to:

- the OAuth2 guide for connecting to PSD2 / regulatory APIs
- the OAuth2 guide for connecting to Premium APIs

OAuth2 guide for connecting to PSD2 APIs

Before you can access ING's APIs, you first need to authenticate your application and retrieve an application- or customer access token. ING authenticates clients with the use of public key cryptography and uses OAuth 2.0 RFC6749 as industry-standard protocol for authorization. The Authentication part is composed of Mutual TLS (typical end-point authentication and encryption) and HTTP Request signing (application level authentication and integrity) which is described in HTTP Signatures Draft RFC version 10. With OAuth 2.0 ING authorizes access to our APIs and also obtains consent/approval of a customer to allow a client to execute an action on behalf of the customer such as a payment request. Not all of this standard is used. ING implemented these functionalities of OAuth 2.0:

- Client credentials flow - allows access to APIs that don't act on behalf of a customer
- Authorization code flow - allows access to APIs that act on behalf of a customer
- Refresh token flow - allows refreshing a previous access token
- Revoke token flow - allows a client to revoke a token.

Note: The authentication mechanism for accessing PSD2 APIs is different. This is based on PSD2 specific eIDAS Qualified Certificates or OBIE Directory Certificates as further elaborated in this documentation.

Prerequisites

Some tasks need to be done before we can go ahead and try out any requests to the ING APIs. First we need two different PKI key pairs. You can use official CA signed certificates, eIDAS or OBIE certificates supporting PSD2. The TLS connection has to be 2-way. Both the client and the server authenticates themselves to each other.

Note: Your application must use TLS 1.3 without CBC (or at least TLS 1.2 without CBC) as this is the minimum security standard required.

Certificate generation

To secure the connection to our APIs, we require TLS Client Authentication and HTTP message signing for both authentication and end-to-end data integrity:

- To call PSD2 APIs, use eIDAS Qualified website authentication certificate (QWAC) or OBIE Qualified website authentication certificate (OBWAC) and eIDAS Qualified seal certificate (QSEAL) or OBIE Qualified seal certificate supporting PSD2 (OBSEAL).

Two X.509 public key certificates are required for authentication:

- A TLS connection certificate used for setting up a mutual TLS connection
- An HTTP Signature certificate used for signing the requests

Request signature

HTTP Request signing

Check the API documentation of the API you want to connect to and determine whether it require

Provide feedback

message signing (and which type of message signing) or whether it requires mTLS (mutual TLS) only. You can jump to the next section of this Get Started guide based on these requirements.

- When the API requires an "X-JWS-Signature" header to be supplied, it requires JWS signing .
- When the API and endpoint requires a "Signature" header to be supplied, it requires HTTP signing .

The newer and latest versions of our PSD2 APIs will require an "X-JWS-Signature" header.

Below we will explain the steps to create a valid signature.

1. Calculate the digest of the body of your request

The digest is the SHA-256 hash value of the requests body encoded in Base64. If the body is empty, it should be the SHA-256 value of the empty string.

Example digest header for an empty body:

```
Digest: SHA-256=47DEQpj8HBSa+/TImW+5JCeuQeRkm5NMpJWZG3hSuFU=
```

2. Set the date header to the current date and time

The Date header requires the current date in the HTTP standard Date header format (see RFC7231).

Important: We enforce a strict time frame for the request. If the Date header's time component diverges by more than 3 minutes, the request will be considered invalid and discarded.

Example Date header:

```
Date: Wed, 03 Jul 2019 08:28:28 GMT
```

3. Create and sign the signing string

For creating the signing string, follow the guide in Section 2.3 of the HTTP Signatures Draft RFC. The minimal set of headers for requests to ING APIs are:

- (request-target)
- Digest
- Date

The request target is a combination of the HTTP action verb and the request path. These are taken from the HTTP request, no additional header is needed.

Important: The individual ING APIs can have additional mandatory headers which are specified in the API documentation of that API.

Example signing string:

```
(request-target): get /greetings/single\

date: Wed, 03 Jul 2019 08:28:28 GMT\

digest: SHA-256=47DEQpj8HBSa+/TImW+5JCeuQeRkm5NMpJWZG3hSuFU=
```

Note: The '\
' symbols above are included to demonstrate where the new line character should be inserted. There is no new line on the final line of the signing string.

To create the signature value, the signing string needs to be signed with your signing key using the RSA-SHA256 or ECDSA-SHA256, ECDSA-SHA384, ECDSA-SHA512 algorithm and Base64 encoded.

4. Set the signature header in the request

The HTTP signature header can now be constructed. The `headers` parameter is used to specify the ordered list of lowercased HTTP headers included when generating the signature for the request.

Example Signature header:

For RSA:

```
Signature: keyId=\"[CLIENT_ID]\",algorithm=\"rsa-sha256\",headers=\"
(request-target) date digest\",signature=\"[SIGNATURE_VALUE]\"
```

For ECDSA:

```
Signature: keyId=\"[CLIENT_ID]\",algorithm=\"ecdsa-sha256\",headers=\"
(request-target) date digest\",signature=\"[SIGNATURE_VALUE]\"
```

For the initial authentication request to get an application access token (Client credentials OAuth 2.0 flow), ING uses the 'Authorization header' with the signature. See section Client Credentials flow of this document for more information.

Tips and common mistakes

- Since the generation of the signature input has to be an exact match, please read and follow the draft specification closely HTTP Signatures Draft RFC version 10
- The header identifiers in the headers parameter must be in lower-case
- During the signature string construction, if the pseudo header (request-target) is used, it's value must be carefully created (lowercase only the :method pseudo-header and not the :path pseudo-header; include also the query parameters into the :path pseudo-header). For more details check Section 2.3 from the draft specification.
- When constructing the signing string, the \
  character is a LF (line-feed) character not a LF CR (line-feed carriage-return, like on Windows)
- Strip any leading or trailing white space from the headers before you use them

JWS Request signing

Please refer to Premium API - Jws Signing

Client credentials flow

Some ING APIs don't act on behalf of the customer and thus don't require customer authorisation to take action. These APIs can use the OAuth2.0 client credentials grant flow (also called "two legged OAuth2.0" or "server-to-server") to generate an application access token.

Important: When accessing **PSD2 APIs with eIDAS or OBIE certificates** supporting PSD2, developer registration, application creation, certificate generation and upload, and subscribing to an API steps are not required. Based on the PSD2 roles in the certificate, an application and API subscriptions will automatically be created when requesting an application access token. Currently, we don't offer the possibility to upload eIDAS or OBIE certificates supporting PSD2 via the ING Developer Portal. We will inform you via this portal as soon as we can.

Authenticate your application to retrieve an application access token

ING's OAuth 2.0 endpoint is https://api.ing.com/oauth2/token using the client_credentials grant_type and the following headers and body, see also the API reference.

Header	Value	Remarks
Content-Type	Value: application/x-www-form-urlencoded	This causes the parameters in the body to use the x-www-form-urlencoded compliant URL encoding: note especially that spaces are encoded as + (and not as %20)
Date	Value: "Wed, 14 Feb 2018 11:15:00 GMT"	HTTP's standard Date header (see RFC7231). Important: We enforce a strict time frame within which the request has to be issued. If the Date header's time component diverges by more than 3 minutes, the request will be considered invalid.
Digest	Value: "SHA-256=[the Base64 encoded SHA256 hash value of the body]"	The Base64 encoded SHA-256 hash of the body. If the body is empty the value is "SHA-256=47DEQpj8HBSa+/TImW+5JCeuQeRkm5NMpJWZG3hSuFU="
	- For RSA: Value: Signature keyId="[Client ID]",algorithm="rsa-sha256",headers="[header names]",signature="	

Authorization	[signature value]" - For ECDSA: Value: Signature keyId="[Client ID]",algorithm="ecdsa-sha256",headers="[header names]",signature="[signature value]"	For more information see the "Authorization signature header" section below, and see: HTTP Signatures Draft RFC version 10
TPP-Signature-Certificate	Value: [Base64 encoded certificate in PEM format]	**Only required when using eIDAS or OBIE certificates** with no registration in the ING Developer portal

HTTP body:

```
grant_type=client_credentials&scope=[SCOPE_TOKENS]
```

A request for an application access tokens can contain multiple scope-tokens in one scope separated by spaces, the scope value should be encoded using x-www-form-urlencoded-compliant URL encoding. The requested scope-tokens should be registered in the Developer Portal. When no scope is requested the application access token will contain all the registered scope-tokens by default.

Note: When using eIDAS or OBIE certificates supporting PSD2 the scope parameter is not required. The scopes will be derived automatically from the PSD2 roles in the certificate

Example encoded body:

```
grant_type=client_credentials&scope=greetings%3Aview
```

Authorization 'signature' header

The authorization signature needs to be sent in the 'authorization' HTTP header as described in the HTTP Signatures Draft RFC version 10 . The minimal required headers in the signature are: "(request-target) date digest".

See section HTTP Request Signing for more details on the HTTP Signature header.

Example of the authorization header:

For RSA:

```
Authorization: Signature keyId=\"[eIDAS or OBIE signing certificate serial number]\",algorithm=\"rsa-sha256\",headers=\"(request-target) date digest\",signature=\"[SIGNATURE_VALUE]\"
```

For ECDSA:

```
Authorization: Signature keyId=\"[eIDAS or OBIE signing certificate serial number]\",algorithm=\"ecdsa-sha256\",headers=\"(request-target) date digest\",signature=\"[SIGNATURE_VALUE]\"
```

Value of keyID for eIDAS and OBIE certificates supporting PSD2

When using eIDAS or OBIE certificates supporting PSD2, the keyId value is of the format: SN=XXX. Where "XXX" is the serial number of the certificate in hexadecimal coding. The serial number can be extracted from your certificate using the openssl command:

```
openssl x509 -in [eIDAS QSEALC or OBIE OBSEALC] -serial -noout
```

Example output:

```
serial=499602D2
```

And the keyID will be:

```
SN=499602D2
```

Example request:

For RSA:

```
POST /oauth2/token
Host: api.ing.com
Date: Sun, 05 Jan 2014 21:31:40 GMT
Content-Type: application/x-www-form-urlencoded
Digest: SHA-256=47DEQpj8HBSa+/TImW+5JCeuQeRkm5NMpJWZG3hSuFU=
Authorization: Signature keyId=\"[eIDAS or OBIE signing certificate serial
number]\",algorithm=\"rsa-sha256\",headers=\"(request-target) date
digest\",signature=\"[SIGNATURE_VALUE]\"

grant_type=client_credentials&scope=greetings%3Aview
```

For ECDSA:

```
POST /oauth2/token
Host: api.ing.com
Date: Sun, 05 Jan 2014 21:31:40 GMT
Content-Type: application/x-www-form-urlencoded
Digest: SHA-256=47DEQpj8HBSa+/TImW+5JCeuQeRkm5NMpJWZG3hSuFU=
Authorization: Signature keyId=\"[eIDAS or OBIE signing certificate serial
number]\",algorithm=\"ecdsa-sha256\",headers=\"(request-target) date
digest\",signature=\"[SIGNATURE_VALUE]\"

grant_type=client_credentials&scope=greetings%3Aview
```

If the request is successful you will get back a response like this:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8

{
 \"access_token\": \"eyJhbGciOiJkaXIiLCJlb........\",
 \"expires_in\": \"900\",
 \"scope\": \"greetings:view\",
 \"token_type\": \"Bearer\",
 \"client_id\": \"ff5d0aa0-95c3-4a9f-8b77-.........\"
 \"keys\": [{
    \"kty\":\"RSA\",
    \"use\": \"sig\",
    \"n\": \"pv5aK1\",
    \"e\":\"AQAB\",
    \"alg\":\"RS256\",
    \"x5t\":\"993b40ddb5f6e13c25ec6dbc3662d2987bc4514d\"
  }]
}
```

Important: The response will contain the client ID of your application, this client ID has to be used in the rest of the session when the client ID or key ID is required.

As an extra security measure, the response of a request made to an ING API endpoint can optionally be signed with the ING public certificate to prevent response modifications depending on the specific API requirements. Therefore, the ING public certificate is part of the response body and uses the JSON Web Key Set format RFC7517 section 4 , so it can be used to validate the response signature of requests. With the application access token ING APIs can be called as described in section Call an ING API with the access token.

Error messages client credentials flow

The error response follows these formats:

```
HTTP/1.1 404 Not Found
Content-Type: application/json;charset=UTF-8
{
  \"message\": \"ClientId [25085501-9918-4c46-9a74-009205a0bc88] could not
be found.\"
}
```

```
HTTP/1.1 401 Unauthorized
Date: Thu, 16 Apr 2020 08:47:10 GMT
```

Full list of error responses:

HTTP Response Code	Error message	Comment
401	Body digest value could not be verified.	We cannot verify the request's digest.
404	CLient ID XXXX Could not be found	The supplied client id was invalid
401	Signature could not be successfully verified.	Either the signature is malformed or the information required for constructing that signature is invalid or erroneous.
401	No valid client certificate.	We could not find a certificate matching the thumbprint of the certificate in our client / relying party registration
400	Validation failed: header 'authorisation' should contain an acceptable algorithm.	This means that the algorithm parameter in the authorisation Signature Header is anything but rsa-sha256 or ecdsa-sha256, ecdsa-sha384, ecdsa-sha512
400	Given scopes were invalid.	This is the scope element of OAuth 2.0 specification c.f.r. https://tools.ietf.org/html/rfc6749#appendix-A.4
400	Requirement failed: Validation failed: Signature Certificate should be an OBIE Certificate of type OBSealC.	OBIE flow: This error is given if the signature certificate (TPP-Signature-Certificate header) is not a valid OBIE OBSEAL certificate (OBIE certificate with purpose signing)
400	Requirement failed: Validation failed: TLS Certificate should be an OBIE Certificate of type OBWAC.	OBIE flow: This error is given if the TLS certificate is not a valid OBIE OBWAC (OBIE certificate with purpose authentication)
400	Requirement failed: Validation failed: Signature Certificate should be an eIDAS Certificate of type QSealC.	eIDAS flow: This error is given if the signature certificate (TPP-Signature-Certificate header) is not a valid eIDAS QSEAL certificate (eIDAS certificate with purpose signing)
400	Requirement failed: Validation failed: TLS Certificate should be an eIDAS Certificate of type QWAC.	eIDAS flow: This error is given if the TLS certificate is not a valid eIDAS QWAC (eIDAS certificate with purpose authentication)
401	Certificate not signed by a trusted issuer.	eIDAS/OBIE flow: The received eIDAS or OBIE certificate is not signed by an eIDAS QTSP (eIDAS Qualified Trust Service Provider) or OBIE (Open Banking Implementation Entity).
401	Unauthorized	request was malformatted or otherwise invalid
400	Bad request	InputValidation failed: Field 'client_id' was not provided.

Access token expiration

The access tokens expire after 900 seconds (15 minutes) after which your application needs to re-authenticate to get a new application access token. This expiration time is always mentioned in the token as: "expires_in": "900" where 900 is in seconds.

Authorization code flow

Some of the ING APIs require authorization from the customer, particularly for PSD2 APIs. In this case you will use the ING OAuth 2 Authorization code grant flow, which will provide your application with specific access and refresh tokens for getting the customer data approved by the customer. This flow is sometimes called "three legged OAuth2".

To start the flow you first need to:

- Register your OAuth2 Redirect URI.
- Ensure that your application is subscribed to the API with the scopes you want to request from the customer
- Obtain an application access token with the "granting" scope to start the OAuth 2.0 Authorization Grant code flow (see the Client Credentials grant type)

All authenticated API requests to https://api.ing.com require the Authorization header with bearer access token and the Signature HTTP header as described in section Call an ING API with the access token of this document.

Step 1: Redirect the customer to ING's authorization application and let the customer authorize your application

The customer authorization process is supported through the ING Authorization web or mobile application. Your application should redirect ING customers to their default browser. ING customers can only perform strong customer authentication in the trusted ING environment if they have the tools provided by their browser, such as the URL bar and Transport Layer Security (TLS) certificate information. For native applications, this means ING authorization page must open in the default browser. Native applications can use custom URL schemes as redirect URIs to redirect the ING customer back from the browser to the application requesting permission.

Important: : Redirect the customer to their default browser in which they can confirm that they are communicating with the trusted ING website. You are not allowed to embed ING's authorization page in your application in any way! ING can block your application from using ING APIs when identifying such an implementation.

Redirect the customer to ING's authorization application URL, by appending your application specific parameters:

```
https://myaccount.ing.com/authorize/v2/[COUNTRY_CODE]?
client_id=[YOUR_CLIENT_ID]
&scope=[SCOPES_SPACE_SEPARATED_AND_URLENCODED]
&state=[SOME_ARBITRARY_BUT_UNIQUE_STRING]
&redirect_uri=[YOUR_URL_ENCODED_REDIRECT_URI]
&response_type=code
```

Country code
The country_code is an optional parameter and allows you to redirect your customer to the specified ING country login page. The format is a two-letter value (ISO 3166-1). If you don't add the country_code, the ING customer will be redirected to an ING country selection page.

Country	Code
Belgium	BE
The Netherlands	NL
Spain	ES
Romania	RO
Luxembourg	LU
Italy	IT
Germany	DE
Wholesale Banking	WB

client_id

Required parameter, the unique client_id obtained when you registered your application with ING.

scope

Required parameter, a list of scopes separated by spaces for which an authorization request is being sent. This could be a subset of the scopes you application is allowed to request. The value must be URL encoded, where spaces may be encoded to either "+" or "%20".

state

Required parameter, can be used by your application to maintain state between HTTP redirects and achieve per-request customization of your redirect URL, please use the "state" parameter as described in the OAuth 2.0 RFC.

The state parameter must not be blank. Allowed characters for the "state" parameter are:

- Alphanumeric characters in range A-Z, a-z, 0-9
- Special characters =%#,;+/\\-_
- Any whitespace character \\r \
  \\t \\f \\v

redirect_uri

Optional parameter, a specific redirect URI where the customer will return to once the authorization process is ended. The value must be URL encoded. This parameter is optional if your application has registered only a single redirect URL with ING. This is a mandatory parameter in case your application has registered multiple redirect URLs and should be one of them.

response_type

Required parameter, for PSD2 flow, the value must be "code".

Example redirect url

An example URL to redirect the customer to start the authorization flow should look like this:

```
https://myaccount.ing.com/authorize/v2/NL?client_id=6414808c-d8da-450e-
b10d-8a1c1dd37561&scope=payment-accounts%3Atransactions%3Aview+payment-
accounts%3Abalances%3Aview&state=123456&response_type=code&redirect_uri=htt
p%3A%2F%2Fapi.example.com
```

Error messages Step 1

Any errors are returned via HTTP Redirect to the specified redirect_uri in the request. The error response follows this format:

```
[YOUR_REDIRECT_URI]?
  error=[ERROR]]
  &error_description=[ERROR_DESCRIPTION]
  &state=[YOUR_STATE_VALUE]
```

Full list of returned error codes:

Error	Error Description
server_error	The authorization server encountered an unexpected condition that prevented it from fulfilling the request
server_error	Request state should not be empty
server_error	The request is missing a required parameter, value is either null or empty
invalid_request	The request is missing a required parameter, includes an invalid parameter value, includes a parameter more than once, or is otherwise malformed

invalid_request	The language is invalid, unknown, or malformed
invalid_request	The request is missing a mandatory header
invalid_request	The request is incomplete
invalid_request	The request contains an invalid header
invalid_request	Missing mandatory means information
invalid_request	The URL you have reached is not in service at this time (404).
invalid_request	There are no permissions for this (party,account,scopes) combination.
invalid_request	The requested redirectUrl is invalid, unknown or malformed
invalid_request	Client registered scopes are invalid, unknown or malformed.
invalid_scope	The requested scope is invalid, unknown, or malformed
unsupported\\_response\\_type	The authorization server does not support obtaining an authorization code using this method
unsupported_grant\\_type	The authorization server does not support this grant type

Step 2: Customer authorizes your application

The ING Authorization application will handle the customer authorization flow, including secure log-in and approval of the requested authorization. The ING Authorization application is a web or mobile application, depending on the country the ING customer is banking in.

Important: The customer can authorize access to maximum 50 accounts in a single authorization flow.

Error messages step 2

They are the same as for step 1.

Step 3: Customer is redirected back to your application

Once successful, the ING Authorization application redirects the customer to the given redirect URI for the application with an authorization code included. For example:

```
[YOUR_REDIRECT_URI]?
  state=[YOUR_STATE_VALUE]
  &code=[ING_AUTHORIZATION_CODE]
```

Error messages step 3

They are the same as for step 2.

Step 4: Retrieve customer access token with the authorization code

The authorization code is only valid for a limited period of time (10 minutes) after issuing. Your application should exchange the authorization code for a pair of access and refresh tokens by calling the /oauth2/token endpoint:

For RSA:

```
POST /oauth2/token
Host: api.ing.com
Date: [DATE]
Content-Type: application/x-www-form-urlencoded
Digest: SHA-256=[DIGEST]
Authorization: Bearer [ACCESS_TOKEN]
Signature: keyId=\"[CLIENT_ID]\",algorithm=\"rsa-sha256\",headers=\"
```

```
(request-target) date digest\",signature=\"[SIGNATURE_VALUE]\"

grant_type=authorization_code&code=[ING_AUTHORIZATION_CODE]&redirect_uri=
[YOUR_URL_ENCODED_REDIRECT_URI]
```

For ECDSA:

```
POST /oauth2/token
Host: api.ing.com
Date: [DATE]
Content-Type: application/x-www-form-urlencoded
Digest: SHA-256=[DIGEST]
Authorization: Bearer [ACCESS_TOKEN]
Signature: keyId=\"[CLIENT_ID]\",algorithm=\"ecdsa-sha256\",headers=\"
(request-target) date digest\",signature=\"[SIGNATURE_VALUE]\"

grant_type=authorization_code&code=[ING_AUTHORIZATION_CODE]&redirect_uri=
[YOUR_URL_ENCODED_REDIRECT_URI]
```

If the request is successful you will receive a response as shown below (note: the values of the expiration fields represent seconds):

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8

{
    \"access_token\":\"2YotnFZFEjr1zCsicMWpAA\",
    \"token_type\":\"access\",
    \"expires_in\":300,
    \"refresh_token\":\"tGzv3JOkF0XG5Qx2TlKWIA\",
    \"refresh_token_expires_in\":3600,
    \"scope\":\"scope1 scope2\",
}
```

The customer access token has an expiration time but can be refreshed with a refresh token. Please be aware that the refresh token must be valid, accompanied by a valid application access token. For more information see: RFC6749 section 4.1.3 .

Important: Once an authorization code has been successfully exchanged it will not be possible to reuse it again, this kind of behaviour will lead to the revocation of the initial access(refresh token)

Error messages step 4

The error response follows this format:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
  \"error\":\"[ERROR]\",
  \"error_description\": \"[ERROR_DESCRIPTION]\"
}
```

Full list of returned error messages:

HTTP Status code	Error	Error Description
400	server_error	The authorisation server encountered an unexpected condition that prevented it from fulfilling the request
400	server_error	Client_Id parameter should not be empty
400	server_error	The request is missing a required parameter, value is either null or empty
		The request is missing a required parameter, includes an invalid parameter

400	invalid_request	value, includes a parameter more than once, or is otherwise malformed
400	invalid_request	authorisation Code should not be empty
400	invalid_request	authorisation code expired
400	invalid_request	Invalid access token
400	invalid_request	The request is missing a mandatory header
400	invalid_request	The request is incomplete
400	invalid_request	The request contains an invalid header
400	invalid_request	The provided authorisation code is invalid
400	invalid_request	The URL you have reached is not in service at this time (404)
400	invalid_grant	The provided authorisation code was issued to another client

Call an ING API with the access token

With the application (client credentials) or customer access token (authorization code) you can now call APIs of ING. All the API requests should be signed using the 'Signature' HTTP header as described in the 'Signing HTTP Messages' RFC draft version 10, and should contain at least the following headers: Date, Digest, Signature, Authorization

The "Authorization" header contains the access token as bearer token, example:

```
Authorization: Bearer [ACCESS_TOKEN]
```

HTTP Signature header

See section HTTP Request Signing for more details on the HTTP Signature header.

Example request

For RSA:

```
GET /greetings/single
Host: api.ing.com
Date: Sun, 05 Jan 2014 21:31:40 GMT
Digest: SHA-256=47DEQpj8HBSa+/TImW+5JCeuQeRkm5NMpJWZG3hSuFU=
Authorization: Bearer [ACCESS_TOKEN]
Signature: keyId=\"[CLIENT_ID]\",algorithm=\"rsa-sha256\",headers=\"
(request-target) date digest\",signature=\"[SIGNATURE_VALUE]\"
```

For ECDSA:

```
GET /greetings/single
Host: api.ing.com
Date: Sun, 05 Jan 2014 21:31:40 GMT
Digest: SHA-256=47DEQpj8HBSa+/TImW+5JCeuQeRkm5NMpJWZG3hSuFU=
Authorization: Bearer [ACCESS_TOKEN]
Signature: keyId=\"[CLIENT_ID]\",algorithm=\"ecdsa-sha256\",headers=\"
(request-target) date digest\",signature=\"[SIGNATURE_VALUE]\"
```

Refresh an access token

Customer access tokens will expire after 5 minutes, in the Authorization code grant flow you will get a refresh token to obtain a new customer access token, so you don't need to do a new authorization request. The refresh token will also expire, the expiration time depends on the subscribed ING API (e.g. 90 days). When the refresh token is expired, you need to start the Authorization code flow again and ask authorization of the ING customer.

You can refresh an access token by calling the /oauth2/token endpoint with a **valid application access token** and with the body:

For RSA:

```
POST /oauth2/token
Host: api.ing.com
Date: [DATE]
Content-Type: application/x-www-form-urlencoded
Digest: SHA-256=[DIGEST]
Authorization: Bearer [ACCESS_TOKEN]
Signature: keyId=\"[CLIENT_ID]\",algorithm=\"rsa-sha256\",headers=\"
(request-target) date digest\",signature=\"[SIGNATURE_VALUE]\"

grant_type=refresh_token&refresh_token=[CUSTOMER_REFRESH_TOKEN]
```

For ECDSA:

```
POST /oauth2/token
Host: api.ing.com
Date: [DATE]
Content-Type: application/x-www-form-urlencoded
Digest: SHA-256=[DIGEST]
Authorization: Bearer [ACCESS_TOKEN]
Signature: keyId=\"[CLIENT_ID]\",algorithm=\"ecdsa-sha256\",headers=\"
(request-target) date digest\",signature=\"[SIGNATURE_VALUE]\"

grant_type=refresh_token&refresh_token=[CUSTOMER_REFRESH_TOKEN]
```

If the request is successful, you will receive a response as shown below (note: the values of the expiration fields represent seconds):

Note: The refresh token can have a maximum lifetime of one year. When a customer authorizes your application for more than one year, you need to refresh the tokens at least once every year. The response will contain a new refresh token which again has a lifetime of one year. When a customer authorizes your application for less than one year, when refreshing the tokens, the response will not contain a new refresh token.

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8

{
    \"access_token\":\"2YotnFZFEjr1zCsicMWpAA\",
    \"token_type\":\"access\",
    \"expires_in\":300,
    \"refresh_token\":\"tGzv3JOkF0XG5Qx2TlKWIA\",
    \"refresh_token_expires_in\":3600,
    \"scope\":\"scope1 scope2\",
}
```

For more information see: RFC6749 section 6 .

Error messages refreshing an access token

Generic error message pattern:

```
HTTP/1.1 400 Bad Request
{
  \"error\":\"[ERROR]\",
  \"error_description\":\"[ERROR_DESCRIPTION]\"
}
```

Authorization has expired:

```
HTTP/1.1 400
{
   \"error\":\"invalid_grant\",
   \"error_description \":\"Refresh token has expired.\"
}
```

Authorization has been revoked by the ING customer:

```
HTTP/1.1 400
{
   \"error\":\"invalid_grant\",
   \"error_description \":\"Refresh token is revoked.\"
}
```

Revoke a refresh token

You can revoke a refresh token (and implicitly the authorization of the customer) by calling the /oauth2/token/revoke endpoint with a valid application access token and with the body:

For RSA:

```
POST /oauth2/token/revoke
Host: api.ing.com
Date: [DATE]
Content-Type: application/x-www-form-urlencoded
Digest: SHA-256=[DIGEST]
Authorization: Bearer [ACCESS_TOKEN]
Signature: keyId=\"[CLIENT_ID]\",algorithm=\"rsa-sha256\",headers=\"
(request-target) date digest\",signature=\"[SIGNATURE_VALUE]\"

token=[CUSTOMER_REFRESH_TOKEN]&token_type_hint=refresh_token
```

For ECDSA:

```
POST /oauth2/token/revoke
Host: api.ing.com
Date: [DATE]
Content-Type: application/x-www-form-urlencoded
Digest: SHA-256=[DIGEST]
Authorization: Bearer [ACCESS_TOKEN]
Signature: keyId=\"[CLIENT_ID]\",algorithm=\"ecdsa-sha256\",headers=\"
(request-target) date digest\",signature=\"[SIGNATURE_VALUE]\"

token=[CUSTOMER_REFRESH_TOKEN]&token_type_hint=refresh_token
```

If the request is successful, you will receive a response with a 200 status code. For more information see: RFC7009 .

Error messages revoking a refresh token

```
HTTP/1.1 400 Bad Request
{
  \"error\":\"[ERROR]\",
  \"error_description\":\"[ERROR_DESCRIPTION]\"
}
```

Full list of returned error messages:

HTTP Status code	Error	Error Description
400	server_error	The authorisation server encountered an unexpected condition that prevented it from fulfilling the request
400	server_error	The request is missing a required parameter, includes an invalid parameter value, includes a parameter more than once, or is otherwise malformed

OAuth2 guide for connecting to Premium APIs

Before you can access ING's APIs, you first need to authenticate your application and retrieve an application access token or customer access token. ING authenticates clients with the use of public key cryptography and uses OAuth 2.0 RFC6749 as industry-standard protocol for authorization. The

Authentication mechanism uses Mutual TLS (typical end-point authentication and encryption). With OAuth 2.0 ING authorizes access to our APIs and also obtains consent/approval of a customer to allow a client to execute an action on behalf of the customer such as a payment request. Not all of this standard is used. ING implemented these functionalities of OAuth 2.0:

- Client credentials flow - allows access to APIs that don't act on behalf of a customer
- Authorization code flow - allows access to APIs that act on behalf of a customer
- Refresh token flow - allows refreshing a previously obtained access token
- Revoke token flow - allows a client to revoke a token.

Prerequisites

ING supports mTLS and message signing to verify the identity of the sender of the message and to verify that the message was not tampered with during transit. Therefore you should upload X.509 public key certificates:

- A TLS connection certificate used for setting up a mutual TLS connection
- Optional: A certificate used for signing your requests to ING. Only needed when using message signing

We allow you to upload two pairs of TLS and signing certificates. You must upload at least one unique TLS certificate. If the API you want to consume requires message signing, then you must also upload at least one unique signing certificate. The second pair is optional and can be used as a fallback for when the first pair of certificates is about to expire. We always verify a request using both the certificate pairs.

You can use official CA signed certificates or use so called 'Self signed' certificates.

Note: Your application must use TLS 1.3 without Cipher-Block-Chaining (CBC) (or at least TLS 1.2 without CBC) as this is the minimum security standard required.

Important: In a previous step, if you have followed the 'Getting Started Guide', you registered your own client application in the Developer Portal. If you did then you will register the certificates that you will generate using the instructions here below on that client. That client cannot be used on the sandbox environment api.sandbox.ing.com but only on production api.ing.com!

Certificate generation

To secure the connection to our APIs, we require TLS Client Authentication for authentication and optionally message signing for end-to-end data integrity:

- To call Premium APIs, use your TLS client certificate and message signing certificate.

Up to two X.509 public key certificates are required for authentication:

- A TLS connection certificate used for setting up a mutual TLS connection
- Optional: A message signing certificate used for signing the requests. Only needed when using message signing.

The self-signed certificates can be generated using OpenSSL:

1. Generate a new RSA or EC private key. The password of the private key can be entered during execution of the command, so it doesn't end up in the history of your shell.

- For RSA: openssl genrsa -out server.key -passout stdin 2048
- For EC: openssl ecparam -genkey -name [P-256 or P-384] -out server.key

2. Generate the X.509 Certificate Signing Request

openssl req -sha256 -new -key server.key -out server.csr

3. Sign the X.509 certificate with your own private key

openssl x509 -req -sha256 -days 365 -in server.csr -signkey server.key -out server_public.crt

Important: These steps need to be run two times, once to generate a TLS connection certificate and the next to generate the message signing certificate. Of course if you happen to have one or more 'official signed by a commercial CA' certificates, feel free to use those. As you need to upload these certificates to the developer portal, the CN does not need to reflect the actual DNS name of your application.

Requirements for the TLS and message signing certificate

- Public key algorithm: RSA-2048 bits or higher, EC-224 bits or higher
- Signature algorithm: SHA-256 bits or higher
- Valid upon receipt

- Certificate lifespan must not exceed 39 months
- TLS and message signing certificates are different

Note: The PKI Certificate of ING is signed by our Root CA which in turn is signed by a commercial CA. This way your TLS implementation can be configured to verify the trust chain of ING's TLS certificate used for setting up the TLS tunnel between your client and ING.

Client credentials flow

Some ING APIs don't act on behalf of the customer and thus don't require customer authorization to take action. These APIs can use the OAuth2.0 client credentials grant flow to generate an application access token.

Authenticate your application to retrieve an application access token

ING's OAuth 2.0 endpoint is https://api.ing.com/oauth2/token using the client_credentials grant_type and the following headers and body, see also the API reference.
HTTP header:

Setting the Content-Type header causes the parameters in the body to use the x-www-form-urlencoded compliant URL encoding: note especially that spaces are encoded as + (and not as %20).

```
Content-Type: application/x-www-form-urlencoded
```

HTTP body:

```
grant_type=client_credentials&client_id=[YOUR_APPLICATION_CLIENT_ID]&scope=
[SCOPE_TOKENS]
```

A request for an application access tokens can contain multiple scope-tokens in one scope separated by spaces, the scope value should be encoded using x-www-form-urlencoded-compliant URL encoding. The requested scope-tokens should be registered in the Developer Portal. When no scope is requested the application access token will contain all the registered scope-tokens by default.

The client id is the identifier for your application as registered in the ING Developer portal.

Example encoded body:

```
grant_type=client_credentials&client_id=e77d776b-90af-4684-bebc-
521e5b2614dd&scope=greetings%3Aview
```

Example request:

```
POST /oauth2/token
Host: api.ing.com
Content-Type: application/x-www-form-urlencoded

grant_type=client_credentials&client_id=e77d776b-90af-4684-bebc-
521e5b2614dd&scope=greetings%3Aview
```

If the request is successful you will get back a response like this:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8

{
 \"access_token\": \"eyJhbGciOiJkaXIiLCJlb........\",
 \"expires_in\": \"900\",
 \"scope\": \"greetings:view\",
 \"token_type\": \"Bearer\",
 \"client_id\": \"ff5d0aa0-95c3-4a9f-8b77-.........\"
 \"keys\": [{
    \"kty\":\"RSA\",
    \"use\": \"sig\",
    \"n\": \"pv5aK1\",
    \"e\":\"AQAB\",
    \"alg\":\"RS256\",
    \"x5t\":\"993b40ddb5f6e13c25ec6dbc3662d2987bc4514d\"
  }]
}
```

Important: The response will contain the client ID of your application, this client ID has to be used in the rest of the session when the client ID (or key ID) is required.

With the application access token ING APIs can be called as described in section Call an ING API with the access token.

Error messages client credentials flow

The error response follows these formats:

```
HTTP/1.1 404 Not Found
Content-Type: application/json;charset=UTF-8
{
  \"message\": \"ClientId [25085501-9918-4c46-9a74-009205a0bc88] could not
be found.\"
}
```

```
HTTP/1.1 401 Unauthorized
Date: Thu, 16 Apr 2020 08:47:10 GMT
```

Full list of error responses:

HTTP Response Code	Error message	Comment
400	Bad request	InputValidation failed: Field 'client_id' was not provided.
400	Validation failed: header 'authorization' should contain an acceptable algorithm.	When using HTTP message signing: this means that the algorithm parameter in the authorization Signature Header is anything but rsa-sha256 or ecdsa-sha256, ecdsa-sha384, ecdsa-sha512
400	Given scopes were invalid.	This is the scope element of OAuth 2.0 specification c.f.r. https://tools.ietf.org/html/rfc6749#appendix-A.4
400	Requirement failed: Validation failed: Signature Certificate should be an OBIE Certificate of type OBSealC.	OBIE flow with HTTP message signing: This error is given if the signature certificate (TPP-Signature-Certificate header) is not a valid OBIE OBSEAL certificate (OBIE certificate with purpose signing)
400	Requirement failed: Validation failed: TLS Certificate should be an OBIE Certificate of type OBWAC.	OBIE flow: This error is given if the TLS certificate is not a valid OBIE OBWAC (OBIE certificate with purpose authentication)
400	Requirement failed: Validation failed: Signature Certificate should be an eIDAS Certificate of type QSealC.	eIDAS flow with HTTP message signing: This error is given if the signature certificate (TPP-Signature-Certificate header) is not a valid eIDAS QSEAL certificate (eIDAS certificate with purpose signing)
400	Requirement failed: Validation failed: TLS Certificate should be an eIDAS Certificate of type QWAC.	eIDAS flow: This error is given if the TLS certificate is not a valid eIDAS QWAC (eIDAS certificate with purpose authentication)
401	Certificate not signed by a trusted issuer.	eIDAS/OBIE flow: The received eIDAS or OBIE certificate is not signed by an eIDAS QTSP (eIDAS Qualified Trust Service Provider) or OBIE (Open Banking Implementation Entity).
401	Body digest value could not be verified.	\tWe cannot verify the request's digest.
401	\tSignature could not be	When using HTTP message signing: either the signature is malformed or the information required for constructing

	successfully verified.	that signature is invalid or erroneous.
401	No valid client certificate.	We could not find a certificate matching the thumbprint of the certificate in our client / relying party registration
401	Unauthorized	Request was malformatted or otherwise invalid
404	Client ID XXXX Could not be found	The supplied client id was invalid

Access token expiration

An access token expires after 900 seconds (15 minutes) after which your application needs to re-authenticate to get a new application access token. This expiration time is always mentioned in the token as: "expires_in": "900" where the value 900 is in seconds.

Authorization code flow

Some of the ING APIs require authorization from the customer. In this case you will use the ING OAuth 2 Authorization code grant flow, which will provide your application with specific access and refresh tokens for getting the customer data approved by the customer.

Prerequisites

- Register your OAuth2 Redirect URI via developer.ing.com
- Ensure that your application is subscribed to the API with the scopes you want to request from the customer
- Obtain an application access token with the "granting" scope to start the OAuth 2.0 Authorization Grant code flow (see the Client Credentials grant type)

All authenticated API requests to https://api.ing.com require the Authorization header with bearer access token. In some cases the API request also requires message signing, check out the message signature requirements as described in The Get Started guide for more information

Step 1: Redirect the customer to ING's authorization application and let the customer authorize your application

The customer authorization process is supported through the ING Authorization web or mobile application. Your application should redirect ING customers to their default browser. ING customers can only perform strong customer authentication in the trusted ING environment if they have the tools provided by their browser, such as the URL bar and Transport Layer Security (TLS) certificate information. For native applications, this means ING authorization page must open in the default browser. Native applications can use custom URL schemes as redirect URIs to redirect the ING customer back from the browser to the application requesting permission.

Important: : Redirect the customer to their default browser in which they can confirm that they are communicating with the trusted ING website. You are not allowed to embed ING's authorization page in your application in any way! ING can block your application from using ING APIs when identifying such an implementation.

Redirect the customer to ING's authorization application URL, by appending your application specific parameters:

```
https://myaccount.ing.com/authorize/v2/[COUNTRY_CODE]?
client_id=[YOUR_CLIENT_ID]
&scope=[SCOPES_SPACE_SEPARATED_AND_URLENCODED]
&state=[SOME_ARBITRARY_BUT_UNIQUE_STRING]
&redirect_uri=[YOUR_URL_ENCODED_REDIRECT_URI]
&response_type=code
```

Country code
The country_code is an optional parameter and allows you to redirect your customer to the specified ING country login page. The format is a two-letter value (ISO 3166-1). If you don't add the country_code, the ING customer will be redirected to an ING country selection page.

Country	Code
Belgium	BE

The Netherlands	NL
Spain	ES
Romania	RO
Luxembourg	LU
Italy	IT
Germany	DE
Wholesale Banking	WB

client_id

Required parameter, the unique client_id obtained when you registered your application with ING.

scope

Required parameter, a list of scopes separated by spaces for which an authorization request is being sent. This could be a subset of the scopes you application is allowed to request. The value must be URL encoded, where spaces may be encoded to either "+" or "%20".

state

Required parameter, can be used by your application to maintain state between HTTP redirects and achieve per-request customization of your redirect URL, please use the "state" parameter as described in the OAuth 2.0 RFC.

The state parameter must not be blank. Allowed characters for the "state" parameter are:

- Alphanumeric characters in range A-Z, a-z, 0-9
- Special characters =%#,;+/\\-_
- Any whitespace character \\r \
  \\t \\f \\v

redirect_uri

Optional parameter, a specific redirect URI where the customer will return to once the authorization process is ended. The value must be URL encoded. This parameter is optional if your application has registered only a single redirect URL with ING. This is a mandatory parameter in case your application has registered multiple redirect URLs and should be one of them.

response_type

Required parameter,the value must be "code".

Example redirect url

An example URL to redirect the customer to start the authorization flow should look like this:

```
https://myaccount.ing.com/authorize/v2/NL?client_id=6414808c-d8da-450e-
b10d-8a1c1dd37561&scope=payment-accounts%3Atransactions%3Aview+payment-
accounts%3Abalances%3Aview&state=123456&response_type=code&redirect_uri=htt
p%3A%2F%2Fapi.example.com
```

Error messages Step 1

Any errors are returned via HTTP Redirect to the specified redirect_uri in the request. The error response follows this format:

```
[YOUR_REDIRECT_URI]?
  error=[ERROR]
  &error_description=[ERROR_DESCRIPTION]
  &state=[YOUR_STATE_VALUE]
```

Full list of returned error codes:

HTTP Response Code	Error	Error Description
500	server_error	The authorization server encountered an unexpected condition that prevented it from fulfilling the request
400	invalid_request	Request state should not be empty
400	invalid_request	The request is missing a required parameter, value is either null or empty
400	invalid_request	The request is missing a required parameter, includes an invalid parameter value, includes a parameter more than once, or is otherwise malformed
400	invalid_request	The language is invalid, unknown, or malformed
400	invalid_request	The request is missing a mandatory header
400	invalid_request	The request is incomplete
400	invalid_request	The request contains an invalid header
400	invalid_request	\tMissing mandatory means information
400	invalid_request	The URL you have reached is not in service at this time (404).
400	invalid_request	There are no permissions for this (party,account,scopes) combination.
400	invalid_request	The requested redirectUrl is invalid, unknown or malformed
400	invalid_request	Client registered scopes are invalid, unknown or malformed.
400	invalid_request	The requested scope is invalid, unknown, or malformed
400	unsupported_response_type	The authorization server does not support obtaining an authorization code using this method
400	unsupported_response_type	The authorization server does not support this grant type

Step 2: Customer authorizes your application

The ING Authorization application will handle the customer authorization flow, including secure log-in and approval of the requested authorization. The ING Authorization application is a web or mobile application, depending on the country the ING customer is banking in.

Error messages step 2

They are the same as for step 1.

Step 3: Customer is redirected back to your application

Once successful, the ING Authorization application redirects the customer to the given redirect URI for the application with an authorization code included. For example:

```
[YOUR_REDIRECT_URI]?
  state=[YOUR_STATE_VALUE]
  &code=[ING_AUTHORIZATION_CODE]
```

Error messages step 3

They are the same as for step 2.

Step 4: Retrieve customer access token with the authorization code

The authorization code is only valid for a limited period of time (10 minutes) after issuing. Your application should exchange the authorization code for a pair of access and refresh tokens by calling the /oauth2/token endpoint:

```
POST /oauth2/token
Host: api.ing.com
Content-Type: application/x-www-form-urlencoded
Authorization: Bearer [ACCESS_TOKEN]

grant_type=authorization_code&code=[ING_AUTHORIZATION_CODE]&redirect_uri=
[YOUR_URL_ENCODED_REDIRECT_URI]
```

If the request is successful you will receive a response as shown below (note: the values of the expiration fields represent seconds):

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8

{
    \"access_token\":\"2YotnFZFEjr1zCsicMWpAA\",
    \"token_type\":\"access\",
    \"expires_in\":300,
    \"refresh_token\":\"tGzv3JOkF0XG5Qx2TlKWIA\",
    \"refresh_token_expires_in\":3600,
    \"scope\":\"scope1 scope2\",
}
```

The customer access token has an expiration time but can be refreshed with a refresh token. Please be aware that the refresh token must be valid, accompanied by a valid application access token. For more information see: RFC6749 section 4.1.3 .

Important: Once an authorization code has been successfully exchanged it will not be possible to reuse it again, this kind of behaviour will lead to the revocation of the refresh token.

Error messages step 4

The error response follows this format:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
  \"error\":\"[ERROR]\",
  \"error_description\": \"[ERROR_DESCRIPTION]\"
}
```

Full list of returned error messages:

HTTP Status code	Error	Error Description
500	server_error	The authorisation server encountered an unexpected condition that prevented it from fulfilling the request
500	server_error	Client_Id parameter should not be empty
500	server_error	The request is missing a required parameter, value is either null or empty
400	invalid_request	The request is missing a required parameter, includes an invalid parameter value, includes a parameter more than once, or is otherwise malformed
400	invalid_request	authorisation Code should not be empty

400	invalid_request	authorisation code expired
400	invalid_request	Invalid access token
400	invalid_request	The request is missing a mandatory header
400	invalid_request	The request is incomplete
400	invalid_request	The request contains an invalid header
400	invalid_request	The provided authorisation code is invalid
400	invalid_request	The URL you have reached is not in service at this time (404)
400	invalid_grant	The provided authorisation code was issued to another client

Call an ING API with the access token

With the application access token (client credentials) or customer access token (authorization code) you can now call APIs of ING. In some cases the API request also requires message signing, check out the message signature requirements as described in the Get Started guide for more information .

The "Authorization" header contains the access token as bearer token, example:

```
Authorization: Bearer [ACCESS_TOKEN]
```

Refresh an access token

Customer access tokens will expire after 5 minutes, in the Authorization code grant flow you will get a refresh token to obtain a new customer access token, so you don't need to do a new authorization request. The refresh token will also expire, the expiration time depends on the subscribed ING API (e.g. 90 days). When the refresh token is expired, you need to start the Authorization code flow again and ask authorization of the ING customer.

Note: The refresh token can have a maximum lifetime of one year. When a customer authorizes your application for more than one year, you need to refresh the tokens at least once every year.
The response will contain a new refresh token which again has a lifetime of one year. When a customer authorizes your application for less than one year, when refreshing the tokens, the response will not contain a new refresh token.

You can refresh an access token by calling the /oauth2/token endpoint with a **valid application access token** and with the body:

```
POST /oauth2/token
Host: api.ing.com
Content-Type: application/x-www-form-urlencoded
Authorization: Bearer [ACCESS_TOKEN]

grant_type=refresh_token&refresh_token=[CUSTOMER_REFRESH_TOKEN]
```

If the request is successful, you will receive a response as shown below (note: the values of the expiration fields represent seconds):

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8

{
    \"access_token\":\"2YotnFZFEjr1zCsicMWpAA\",
    \"token_type\":\"access\",
    \"expires_in\":300,
    \"refresh_token\":\"tGzv3JOkF0XG5Qx2TlKWIA\",
    \"refresh_token_expires_in\":3600,
    \"scope\":\"scope1 scope2\",
}
```

For more information see: RFC6749 section 6 .

Error messages refreshing an access token

Generic error message pattern:

```
HTTP/1.1 400 Bad Request
{
  \"error\":\"[ERROR]\",
  \"error_description\":\"[ERROR_DESCRIPTION]\"
}
```

Authorization has expired:

```
HTTP/1.1 400
{
   \"error\":\"invalid_grant\",
   \"error_description \":\"Refresh token has expired.\"
}
```

Authorization has been revoked by the ING customer:

```
HTTP/1.1 400
{
   \"error\":\"invalid_grant\",
   \"error_description \":\"Refresh token is revoked.\"
}
```

Revoke a refresh token

You can revoke a refresh token (and implicitly the authorization of the customer) by calling the /oauth2/token/revoke endpoint with a valid application access token and with the body:

```
POST /oauth2/token/revoke
Host: api.ing.com
Content-Type: application/x-www-form-urlencoded
Authorization: Bearer [ACCESS_TOKEN]

token=[CUSTOMER_REFRESH_TOKEN]&token_type_hint=refresh_token
```

If the request is successful, you will receive a response with a 200 status code. For more information see: RFC7009 .

Error messages revoking a refresh token

```
HTTP/1.1 400 Bad Request
{
  \"error\":\"[ERROR]\",
  \"error_description\":\"[ERROR_DESCRIPTION]\"
}
```

Full list of returned error messages:

HTTP Status code	Error	Error Description
400	server_error	The authorisation server encountered an unexpected condition that prevented it from fulfilling the request
400	server_error	The request is missing a required parameter, includes an invalid parameter value, includes a parameter more than once, or is otherwise malformed

HTTP Request signing DEPRECATED

While some APIs are currently still using HTTP Signatures, ING is standardizing on the use of JWS as per the Open Banking Europe specification [link]. Since ING is deprecating the use of HTTP Signatures, this section is only here for reference during the transition period.

In case you still need to call endpoints of the OAuth API using HTTP Signatures in the transition period, below we will explain the steps and provide additional information to create a valid HTTP Signature when using the Client credentials flow, the Authorization code flow, Refreshing an access token and Revoke a refresh token.

For reference check the HTTP Signatures Draft RFC version 10 .

General steps to create a HTTP Signature

1. Calculate the digest of the body of your request

The digest is the SHA-256 hash value of the requests body encoded in Base64. If the body is empty, it should be the SHA-256 value of the empty string.

Example digest header for an empty body:

```
Digest: SHA-256=47DEQpj8HBSa+/TImW+5JCeuQeRkm5NMpJWZG3hSuFU=
```

2. Set the date header to the current date and time

The Date header requires the current date in the HTTP standard Date header format (see RFC7231).

Important: We enforce a strict time frame for the request. If the Date header's time component diverges by more than 3 minutes, the request will be considered invalid and discarded.

Example Date header:

```
Date: Wed, 03 Jul 2019 08:28:28 GMT
```

3. Create and sign the signing string

For creating the signing string, follow the guide in Section 2.3 of the HTTP Signatures Draft RFC. The minimal set of headers for requests to ING APIs are:

- (request-target)
- Digest
- Date

The request target is a combination of the HTTP action verb and the request path. These are taken from the HTTP request, no additional header is needed.

Important: The individual ING APIs can have additional mandatory headers which are specified in the API documentation of that API.

Example signing string:

```
(request-target): get /greetings/single\

date: Wed, 03 Jul 2019 08:28:28 GMT\

digest: SHA-256=47DEQpj8HBSa+/TImW+5JCeuQeRkm5NMpJWZG3hSuFU=
```

Note: The '\
' symbols above are included to demonstrate where the new line character should be inserted. There is no new line on the final line of the signing string.

To create the signature value, the signing string needs to be signed with your signing key using the RSA-SHA256 or ECDSA-SHA256, ECDSA-SHA384, ECDSA-SHA512 algorithm and Base64 encoded.

4. Set the signature header in the request

The HTTP signature header can now be constructed. The `headers` parameter is used to specify the ordered list of lowercased HTTP headers included when generating the signature for the request.

Example Signature header:

For RSA:

```
Signature: keyId=\"[CLIENT_ID]\",algorithm=\"rsa-sha256\",headers=\"
(request-target) date digest\",signature=\"[SIGNATURE_VALUE]\"
```

For ECDSA:

```
Signature: keyId=\"[CLIENT_ID]\",algorithm=\"ecdsa-sha256\",headers=\"
(request-target) date digest\",signature=\"[SIGNATURE_VALUE]\"
```

For the initial authentication request to get an application access token (Client credentials OAuth 2.0 flow), ING uses the 'Authorization header' with the signature. See section Client Credentials flow of this document for more information.

Tips and common mistakes

- Since the generation of the signature input has to be an exact match, please read and follow the draft specification closely HTTP Signatures Draft RFC version 10
- The header identifiers in the headers parameter must be in lower-case
- During the signature string construction, if the pseudo header (request-target) is used, its value must be carefully created (lowercase only the :method pseudo-header and not the :path pseudo-header; include also the query parameters into the :path pseudo-header). For more details check Section 2.3 from the draft specification.
- When constructing the signing string, the \
character is a LF (line-feed) character not a LF CR (line-feed carriage-return, like on Windows)
- Strip any leading or trailing white space from the headers before you use them

Response signing

As an extra security measure, the response of a request made to an ING API endpoint can optionally be signed with the ING public certificate to prevent response modifications depending on the specific API requirements. Therefore, the ING public certificate is part of the response body and uses the JSON Web Key Set format RFC7517 section 4 , so it can be used to validate the response signature of requests.

Client credentials flow

Authenticate your application to retrieve an application access token

Header	Value	Remarks
Content-Type	Value: application/x-www-form-urlencoded	This causes the parameters in the body to use the x-www-form-urlencoded compliant URL encoding: note especially that spaces are encoded as + (and not as %20)
Date	Value: "Wed, 14 Feb 2018 11:15:00 GMT"	HTTP's standard Date header (see RFC7231). Important: We enforce a strict time frame within which the request has to be issued. If the Date header's time component diverges by more than 3 minutes, the request will be considered invalid.
Digest	Value: "SHA-256=[the Base64 encoded SHA256 hash value of the body]"	The Base64 encoded SHA-256 hash of the body. If the body is empty the value is "SHA-256=47DEQpj8HBSa+/TImW+5JCeuQeRkm5NMpJWZG3hSuFU="
Authorization	- For RSA: Value: Signature keyId="[Client ID]",algorithm="rsa-sha256",headers="[header names]",signature="[signature value]" - For ECDSA: Value: Signature keyId="[Client ID]",algorithm="ecdsa-sha256",headers="[header names]",signature="[signature value]"	For more information see the "Authorization signature header" section below, and see: HTTP Signatures Draft RFC version 10

Authorization 'signature' header

The authorization signature needs to be sent in the 'authorization' HTTP header as described in the HTTP Signatures Draft RFC version 10 . The minimal required headers in the signature are: "(request-target) date digest".

See section HTTP Request Signing for more details on the HTTP Signature header.

Example of the authorization header:

For RSA:

```
Authorization: Signature keyId=\"[CLIENT_ID]\",algorithm=\"rsa-
sha256\",headers=\"(request-target) date digest\",signature=\"
[SIGNATURE_VALUE]\"
```

For ECDSA:

```
Authorization: Signature keyId=\"[CLIENT_ID]\",algorithm=\"ecdsa-
sha256\",headers=\"(request-target) date digest\",signature=\"
[SIGNATURE_VALUE]\"
```

Example request:

For RSA:

```
POST /oauth2/token
Host: api.ing.com
Date: Sun, 05 Jan 2014 21:31:40 GMT
Content-Type: application/x-www-form-urlencoded
Digest: SHA-256=47DEQpj8HBSa+/TImW+5JCeuQeRkm5NMpJWZG3hSuFU=
Authorization: Signature keyId=\"[CLIENT_ID]\",algorithm=\"rsa-
sha256\",headers=\"(request-target) date digest\",signature=\"
[SIGNATURE_VALUE]\"

grant_type=client_credentials&scope=greetings%3Aview
```

For ECDSA:

```
POST /oauth2/token
Host: api.ing.com
Date: Sun, 05 Jan 2014 21:31:40 GMT
Content-Type: application/x-www-form-urlencoded
Digest: SHA-256=47DEQpj8HBSa+/TImW+5JCeuQeRkm5NMpJWZG3hSuFU=
Authorization: Signature keyId=\"[CLIENT_ID]\",algorithm=\"ecdsa-
sha256\",headers=\"(request-target) date digest\",signature=\"
[SIGNATURE_VALUE]\"

grant_type=client_credentials&scope=greetings%3Aview
```

Authorization code flow

Step 1: Get redirect URL to the ING authorization application

For RSA:

```
GET /oauth2/authorization-server-url?scope=
[SCOPES_SPACE_SEPARATED_AND_URLENCODED]&country_code=
[COUNTRY_CODE]&redirect_uri=[YOUR_REDIRECT_URI]
Host: api.ing.com
Date: [DATE]
Digest: SHA-256=[DIGEST]
Authorization: Bearer [ACCESS_TOKEN]
Signature: keyId=\"[CLIENT_ID]\",algorithm=\"rsa-sha256\",headers=\"
(request-target) date digest\",signature=\"[SIGNATURE_VALUE]\"
```

For ECDSA:

```
GET /oauth2/authorization-server-url?scope=
[SCOPES_SPACE_SEPARATED_AND_URLENCODED]&country_code=
[COUNTRY_CODE]&redirect_uri=[YOUR_REDIRECT_URI]
Host: api.ing.com
Date: [DATE]
Digest: SHA-256=[DIGEST]
Authorization: Bearer [ACCESS_TOKEN]
Signature: keyId=\"[CLIENT_ID]\",algorithm=\"ecdsa-sha256\",headers=\"
(request-target) date digest\",signature=\"[SIGNATURE_VALUE]\"
```

Step 5: Retrieve customer access token with the authorization code

For RSA:

```
POST /oauth2/token
Host: api.ing.com
Date: [DATE]
Content-Type: application/x-www-form-urlencoded
Digest: SHA-256=[DIGEST]
Authorization: Bearer [ACCESS_TOKEN]
Signature: keyId=\"[CLIENT_ID]\",algorithm=\"rsa-sha256\",headers=\"
(request-target) date digest\",signature=\"[SIGNATURE_VALUE]\"

grant_type=authorization_code&code=[ING_AUTHORIZATION_CODE]&redirect_uri=
[YOUR_URL_ENCODED_REDIRECT_URI]
```

For ECDSA:

```
POST /oauth2/token
Host: api.ing.com
Date: [DATE]
Content-Type: application/x-www-form-urlencoded
Digest: SHA-256=[DIGEST]
Authorization: Bearer [ACCESS_TOKEN]
Signature: keyId=\"[CLIENT_ID]\",algorithm=\"ecdsa-sha256\",headers=\"
(request-target) date digest\",signature=\"[SIGNATURE_VALUE]\"

grant_type=authorization_code&code=[ING_AUTHORIZATION_CODE]&redirect_uri=
[YOUR_URL_ENCODED_REDIRECT_URI]
```

Refresh an access token

For RSA:

```
POST /oauth2/token
Host: api.ing.com
Date: [DATE]
Content-Type: application/x-www-form-urlencoded
Digest: SHA-256=[DIGEST]
Authorization: Bearer [ACCESS_TOKEN]
Signature: keyId=\"[CLIENT_ID]\",algorithm=\"rsa-sha256\",headers=\"
(request-target) date digest\",signature=\"[SIGNATURE_VALUE]\"

grant_type=refresh_token&refresh_token=[CUSTOMER_REFRESH_TOKEN]
```

For ECDSA:

```
POST /oauth2/token
Host: api.ing.com
Date: [DATE]
Content-Type: application/x-www-form-urlencoded
Digest: SHA-256=[DIGEST]
Authorization: Bearer [ACCESS_TOKEN]
Signature: keyId=\"[CLIENT_ID]\",algorithm=\"ecdsa-sha256\",headers=\"
(request-target) date digest\",signature=\"[SIGNATURE_VALUE]\"

grant_type=refresh_token&refresh_token=[CUSTOMER_REFRESH_TOKEN]
```

Revoke a refresh token

For RSA:

```
POST /oauth2/token/revoke
Host: api.ing.com
Date: [DATE]
Content-Type: application/x-www-form-urlencoded
Digest: SHA-256=[DIGEST]
Authorization: Bearer [ACCESS_TOKEN]
Signature: keyId=\"[CLIENT_ID]\",algorithm=\"rsa-sha256\",headers=\"
(request-target) date digest\",signature=\"[SIGNATURE_VALUE]\"

token=[CUSTOMER_REFRESH_TOKEN]&token_type_hint=refresh_token
```

For ECDSA:

```
POST /oauth2/token/revoke
Host: api.ing.com
Date: [DATE]
Content-Type: application/x-www-form-urlencoded
Digest: SHA-256=[DIGEST]
Authorization: Bearer [ACCESS_TOKEN]
Signature: keyId=\"[CLIENT_ID]\",algorithm=\"ecdsa-sha256\",headers=\"
(request-target) date digest\",signature=\"[SIGNATURE_VALUE]\"

token=[CUSTOMER_REFRESH_TOKEN]&token_type_hint=refresh_token
```

[Premium API - Jws Signing] : https://developer.ing.com/openbanking/resources/get-started/premium#jws