SwissID 5.2.0

Integration Guidelines – OpenID Connect

Release on October 15th, 2024

Table of Content

CHANGELOG

Version	Date	Changes
4.6.0	15.06.2022	Claims and scopes table updated. New scopes – photoid_selfie and identity_documents added. For identity_documents new claims. changelog restored.
4.7.0	11.10.2022	Adding new claim to individual claim urn:swissid:place_of_origin. Updated Profile scope with additional claims. Check and update tick boxes for claims, that allowed in Profile, identity document scopes.
4.9.0	20.02.2022	Adding section EPD step up, with a description of how to add EPD step up URL
4.9.2	4.05.2023	Changes for identity_updated_at
4.10.0	23.05.2023	Remove QOR3 change QOR2 definition, fix typo in step up description, (remove “+” sign)
4.10.2	24.07.2023	Add mandatory acr_values parameter for users, who has lot>0, and add end points to the section ID Document Service
4.13.0	15.01.2024	Requesting QoR2 example fixed
4.13.1	04.03.2024	LoT1 step-up replaced with LoT Step-up with multiple purposes
4.14.0	13.03.2024	Manual checker: splitting up of the queues, PXL SDK upgrade, Security tickets addressed
4.15.0	21.05.2024	New EPD stepup flow (EPD Step Up Phase 3) & Deeplinking in SwissID, EPD stepup improvements (signing request triggered by manual checker)
5.0.0	03.07.2024	Upgrade from Version 6.5 (which is end of life) to 7.4
5.0.3	23.07.2024	SMS abuse protection
5.1.0	10.07.2024	VC, Deffered Deeplinking
5.2.0	15.10.2024	Remote identity verification with web flow

INTRODUCTION

This document provides Relying Parties (RPs) with technical guidance and best practices to integrate their application with the SwissID OpenID Provider (SwissID OP).

OIDC: OpenID Connect

SwissID OAuth 2.0 APIs can be used for both authorization and authentication, which complies with OpenID Connect specification (OIDC).
OIDC is a thin layer on top of the OAuth 2.0 protocol for identification and authentication purposes. It enables Clients to verify the identity of the Identity Owner (IdO) based on the authentication performed by an Authorization Server, as well as to obtain basic profile information about the IdO in an interoperable and REST-like manner (see https://openid.net/specs/openid-connect-core-1_0.html for more details).

OIDC performs authentication to login the IdO or to determine that the IdO is already logged in. OIDC returns the result of the Authentication performed by the Server (SwissID OP) to the Client in a secure manner so that the Client can rely on it. For this reason, the Client is called Relying Party (RP) in this case.

Environments

SwissID IdP provides Pre-production and Production environments:

Environment	Domain: <ENV>	Source IP	Description
Pre-production	https://<ingress>.sandbox.pre.swissid.ch	91.194.146.72	RP integration environment that exactly resembles a production environment. Access should be whitelisted by incoming IP ranges.
Production	https://<ingress>.swissid.ch	91.194.146.64	The production environment is also known *as live*.

Mostly the ingress is `login`, e.g., https://login.swissid.ch, but it could also be `account`, e.g., https://account.swissid.ch. To differentiate them in we use <a.ENV> for when the ingress is `account` and <ENV> when it is `login`.

Reference Application

SwissID provides the OIDC Reference App, where the RPs can test the IdP behavior before and during the integration process:

```
https://rp.sandbox.pre.swissid.ch/oidc/
```

Please note that the Reference App is intentionally disabled for the Production environment.

Level of Trust concept

SwissID defines the Level of Trust (LoT) concept, which reflects how trustworthy an identity may be interpreted by the RPs. SwissID covers the different needs of the identity receiver, the RP, as well as the interests of the IdO by offering different quality levels for the identity following the legislations and standards listed in Compliance sub-section.

SwissID LoT-concept has two dimensions:

1. Quality of Registration (QoR): the quality of the verification on the identity during the identity registration. Ranging from qor0 (self-declared) to qor2 (ID document verification using NFC, or manually checked on-line by RAOs)
2. Quality of Authentication (QoA): the quality of the methods used to authenticate (i.e., authenticators) the IdO. Ranging from qoa1 (username/password) to qoa2 (two-factor authentication, like mTAN or SwissID mobile application).

Hence, the combinations of this two dimensions results in the respective LoT.

	QoA 1 *One-Factor Authentication (1FA)* *password*	**QoA 2** *Two-Factor Authentication (2FA)* *mTAN or SwissID mobile app*
QoR 0 *Self-declared* *without evidence verification*	**LoT 0**	**LoT 0**
QoR 1 *Online presence* *with ID document verification*	*Not Allowed*	**LoT 1**
QoR 2 *ID document verification using NFC, or* *manually checked on-line by RAOs*	*Not Allowed*	**LoT 2**

Furthermore, QoR will also affect which ID attributes are available – please check Scopes and claims section for more details.

Compliance

SwissSign follows international and national laws and recommendations for digital identities. The most important references used in defining the LoT-concept are described in the following table:

Standard	Reference link
NIST SP 800-63-3	https://pages.nist.gov/800-63-3
ISO/IEC 29115:2013	https://www.iso.org/standard/45138.html
eCH-0170 V2.0	https://www.ech.ch/de/standards/60593
eCH-171	https://www.ech.ch/de/standards/60603
EPRA / EPDG	https://www.admin.ch/opc/de/classified-compilation/20111795/index.html
EIDAS	https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32014R0910

	https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32015R1502

GETTING STARTED

To use OIDC for authentication, you will have to have an OIDC client set on SwissID before you can perform the integration. From an OAuth point of view, an OIDC RP is an OAuth Client. Each RP is assigned a unique Client Identifier.

OIDC Discovery

SwissID IdP supports OIDC Discovery which makes a JSON document available at the path formed by concatenating the string /.well-known/openid-configuration to the Issuer, i.e.:

```
<ENV>/idp/oauth2/.well-known/openid-configuration
```

Also, the JSON Web Key Sets (jwks) endpoint can be found at:

```
<ENV>/idp/oauth2/connect/jwk_uri
```

RP Registration

The RP's Client Identifier and a secret are provided to the RP during the on-boarding process. The credentials are sent to the registered email account, e.g.:

Dear Customer,

The online service <CLIENT_ID> has been successfully registered.

Below is your account information summary:

Client ID: <CLIENT_ID>
Environment: <ENV>/idp/oauth2/.well-known/openid-configuration
Password: <CLIENT_SECRET>

(…)

Example of authentication request:

GET "<ENV>/idp/oauth2/authorize?response_type=code&client_id=<CLIENT_ID>&scope=openid%20profile&redirect_uri=<RP_CALLBACK_URL_ENCODED>&nonce=<NONCE>&state=<STATE>&acr_values=loa-1&ui_locales=de"

Example of access token request:

```
POST --header "Authorization: Basic base64EncodingOf(<CLIENT_ID>:<CLIENT_SECRET>)" --data
"grant_type=authorization_code&code=<AUTH_CODE>&redirect_uri=<CALLBACK_URL_ENCODED>"
"<ENV>/idp/oauth2/access_token"
```

Example of userinfo request:

```
GET --header "Authorization: Bearer <ACCESS_TOKEN>" "<ENV>/idp/oauth2/userinfo"
```

If you have any further questions, please contact our technical customer service by e-mail rp-support@swissid.ch.

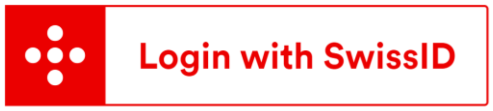
Kind regards,

Your SwissID team

If you did not receive any similar email, means that your on-boarding process is not ready yet, and you need to request your credentials at scc@swisssign.com.

Authorization code grant flow

The authorization URI is a link provided to an IdO on a web page for clicking on for starting the OIDC authorization flow. This is the first step for a RP retrieving the IdO's personal data.



Furthermore, SwissID urges the integrating RPs to not fill in the HTTP Referrer header. Typically, the Referrer header is populated with the address of the page where the request originated, which should not occur due data protection best practices. To do so, SwissID recommends that the HTML hyperlinks have the `rel` attribute set to "`noreferrer`", e.g.:

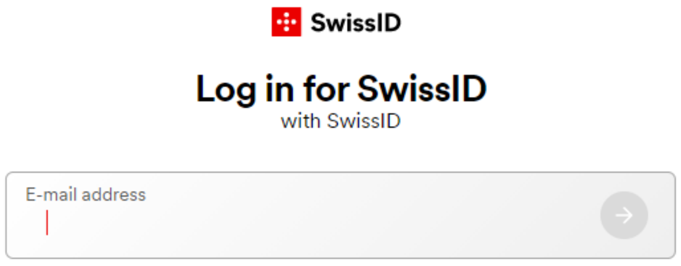
```
<a href="<ENV>/idp/oauth2/authorize?..." rel="noreferrer">link</a>
```

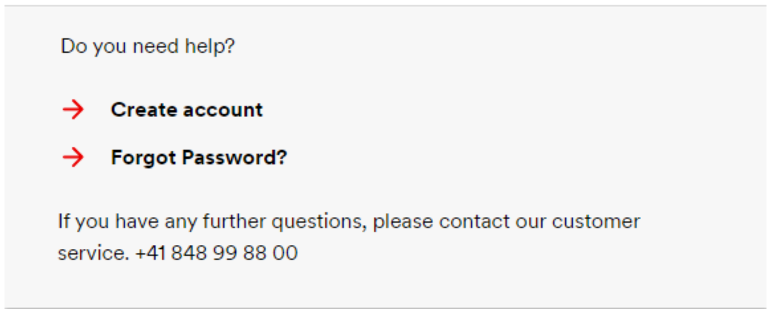
As mentioned before, the RP can trigger the authorization code flow by calling, e.g.:

Authorize endpoint

```
<ENV>/idp/oauth2/authorize?response_type=code&client_id=<CLIENT_ID>&redirect_uri=<RP_CALLBACK_UR
L_ENCODED>&scope=openid
```

In the example above, it is being requested only the `openid` scope and the redirect URI was previously defined in the onboarding step. Hence, only registered URI will be considered for redirecting the IdO back to the RP. After the browser call, the IdO is redirected to SwissID page and should authenticate or create a new SwissID account.




Do you need help?

Create account

Forgot Password?

If you have any further questions, please contact our customer service. +41 848 99 88 00

By pressing the consent button, the IdO is indirectly triggering the callback containing the authorization code bound to the IdO or an error message in the URL callback, e.g.:

```
<RP_CALLBACK_URL_ENCODED>?error=unmet_authentication_requirements&error_description=Requested%20
QoR%20is%20not%20available
```

In the above example, the RP requested a QoR that it is not available for the target IdO, for these cases, SwissID suggests that the RP redirects the IdO to the LoT step-up flow (see IdO LoT step-up for more details).

Other possible errors:

Error	Description
access_denied	The user gave no consent to the scopes and/or Claims requested.
authentication_cancelled	The user cancelled the login process.
interaction_required	The Authentication Request contained the parameter "prompt=none" and the End-User is not yet authenticated.
unmet_authentication_requirements	Requested QoR is not available. unmet_authentication_requirements error is only triggered if the RP was configured without the “Best Effort identity” flag, which will deliver an identity with lower LoT if the requested LoT it is not available. If you want to toggle this functionality, contact SwissID support.

Finally, the **swissid cookie** will also be delivered to the calling browser.

Token endpoint

When a valid call it is performed to the Authorize endpoint and the IdO consents sharing her/his identity data, a callback is sent by the IdP to the redirect URI (<RP_CALLBACK_URL_ENCODED>). The callback will contain the authorization code necessary to retrieve the access_token, refresh_token and id_token, e.g.:

```
<RP_CALLBACK_URL_ENCODED>?code=<AUTH_CODE>&iss=https%3A%2F%2Flogin.
swissid.ch%3A443%2Fidp%2Foauth2&client_id=<CLIENT_ID>
```

With the authorization code it is now possible to exchange it for the access_token and the id_token by calling the token endpoint, e.g.:

```
curl -X POST '<ENV>/idp/oauth2/access_token'
-H 'Authorization: Basic <BASE64(CLIENT_ID:CLIENT_SECRET)>'
-H 'Content-Type: application/x-www-form-urlencoded; charset=UTF-8'
--data-raw
'grant_type=authorization_code&code=<AUTH_CODE>&redirect_uri=<RP_CALLBACK_URL_ENCODED>'
```

The `<CLIENT_ID>` and the `redirect_uri` parameters specified in this call must match those used as part of the authorization code request, or SwissID will not accept the code.
After a successful call, the following JSON object is received, e.g.:

```
{
      "access_token": "<ACCESS_TOKEN>",
      "refresh_token": "<REFRESH_TOKEN>",
      "scope": "openid",
      "id_token": "<ID_TOKEN>",
      "token_type": "Bearer",
      "expires_in": 720
}
```

The id_token will have "urn:swissid:qor" attribute populated. The RP should confirm if the QoR value in the id_token is enough for its LoT requirements. Otherwise, the RP should redirect the IdO to LoT step-up flow (see IdO LoT step-up for more details).
Furthermore, the token endpoint shall be also used to refresh all the above tokens. By adjusting the URL accordantly:

```
curl -X POST '<ENV>/idp/oauth2/access_token'
-H 'Authorization: Basic <BASE64(CLIENT_ID:CLIENT_SECRET)>'
--data-raw 'grant_type=refresh_token&scope=<SCOPES>&refresh_token=<REFRESH_TOKEN>'
```

Userinfo endpoint

Finally, it is possible to request additional claims about the IdO from SwissID. When requesting claims, provide an access token granted in an OIDC flow as an authorization bearer header. The endpoint **will return the claims associated with the scopes granted when the access token was requested.** E.g.:

```
curl '<ENV>/idp/oauth2/userinfo' -H 'Authorization: Bearer <ACCESS_TOKEN>'
```

Which will retrieve a JWT object like:

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJzOXJEbmlnak8vNDVxSEgyZjN0LzFEMWplNWFMNlY5dVlIWmN
0VGxRaUd3PSIsImdlbmRlciI6Im1hbGUiLCJ1cGRhdGVkX2F0IjoxNjA5NDc2MTYyLCJpc3MiOiJodHRwczovL2xvZ2luLm5
pZ2h0bHkuc3dpc3NpZC54eXo6NDQzL2lkcC9vYXV0aDIiLCJsYW5ndWFnZSI6ImRlX0NIIiwiZ2l2ZW5fbmFtZSI6Ikx1aXM
iLCJmYW1pbHlfbmFtZSI6IlJpYmVpcm8iLCJmb3JnZXJvY2siOnsic2lnIjoiL0kkMD93SjZ2bikrWGczN15qNEtHMX13U1l
kckNaUEI-REAtJExZZCJ9fQ.PDfuJN2NifuAwISP9XQkKRgGjh0Gvd3LfrCLmrfPWas
```

You can decode the JWT at https://jwt.io/

Building the authorization URI

The authorization URI can be customized to trigger different behaviors or outcomes in the authentication of an IdO. The parameters are passed via query string when calling the authorize endpoint. In this section we specify the possible values for every parameter.

To expedite the understanding on what parameters can be requested and how SwissID IdP behaves to those requests, SwissID provides the OIDC Reference App:

```
https://rp.sandbox.pre.swissid.ch/oidc/
```

Please note that the Reference App is intentionally not available for the Production environment.

Basic authorization URI request

- `client_id`: <CLIENT_ID> received at the
- RP Registration step.
- `response_type`: code is the only value allowed. (`response_type=code`)
- `scope`: openid is the only mandatory scope, see Scopes and claims section for the remaining possible scopes and claims.
- `redirect_uri`: <RP_CALLBACK_URL_ENCODED> provided by the RP during on-boarding.
- `state`: A surrogate string provided and managed by the RP that will be included in the returned callback as a URI parameter. The state it is typically used for correlating requests and responses. Because the RP's redirect_uri can be guessed, using a state value can increase RP's assurance that an incoming connection is the result of an authentication request initiated by the RP's app. Furthermore, the generated string can be random, or hash

encode of some client state (e.g., a cookie). In this state hash, the RP can validate the response to additionally ensure that the request and response originated in the same browser, mitigating for instance cross-site request forgery.

- `nonce`: A random string provided and managed by the RP for binding a session with an ID token, inhibiting replay attacks. SwissID IdP will place the received nonce in the respective ID token. RPs must verify that the nonce claim Value in the ID token is equal to the value of the nonce parameter sent in the Authentication Request.
- `prompt`: login and consent are the only values allowed. The IdO will be forced to authenticate even if it has an active SwissID session when `prompt=login`. Likewise, if `prompt=consent` it is requested, the IdO will be forced to provide consent again.
- `acr_value`: this could be qoa1 or qoa2, which requests the IdOs to authenticate with a quality of authentication 1 (one-factor) or quality of authentication 2 (two factor). **Please note** that if you are requesting qor1 or qor2 (user lot level greater than 0) you must request acr_values=qoa2
- `ui_locale`: defines language to be displayed in the SwissID authentication pages. Possible values `de` for German, `fr` for French, `it` for Italian and `en` for English.

Scopes and claims

Claims are assertions on the subject (i.e., IdO), made by the IdO (self-declared identities) or by other entities like the Registration Authority Officer (RAO). Scopes can be seen as a set of claims that the RP can request.
The following table lists the scopes/claims the RPs can request from the SwissID OP. The requesting scopes determine which claims (data) it is retrieved from the subsequent token and userinfo endpoints.

The RP can specify the quality of registration required on the IdO's ID in the individual claim `qor.` Furthermore, the presence of some attributes in the responses (i.e., `id_token` and userinfo) depends on the requested `qor` as well. For instance, the claim request:

```
claims={"userinfo":{"urn:swissid:qor":{"value":"qor2"}}}
```

will add to the id_token the attribute `urn:swissid:qor.` Please note that if the authorization call does not specify the claim "urn:swissid:qor" and target value, the id_token won't have the attribute. Furthermore, notice that some claims can only be requested for higher QoRs, e.g. "urn:siwssid:identity_valid_until":

```
{"userinfo":{"urn:swissid:qor":{"value":"qor2"}}, "urn:swissid:identity_valid_until":null}
```

RP can request individual claims directly outside the scopes, note that swissid specific claims can be requested directly in claims request object, and they are returned in both: `userinfo` and `id_token`.

The table below shows all supported claims with assignment to scopes and needed `qor`.

Scope	Claims	~~qor~~	qor 0	qor 1	qor 2	Description
openid *(Mandatory)*	sub	✓	✓	✓	✓	Retrieves the IdO's sub attribute in the ID Token and userinfo
	acr	✓	✓	✓	✓	Quality of authentication performed, e.g.: qoa1
	updated_at	✓	✓	✓	✓	ID last updated at (UTC)
	auth_time	✓	✓	✓	✓	Represents the time (UTC) after the authentication took place and not the time the user gave consent for the requested scopes. Therefore, the auth_time represents the time just before the consent screen is displayed to the IdO.
profile	given_name	✓	✓	✓	✓	(all given names)
	family_name	✓	✓	✓	✓	
	urn:swissid:qor		✓	✓	✓	Quality of registration process, e.g., qor2 (physical presence)
	urn:swissid:first_name			✓	✓	All IdO's verified given names.
	gender	✓	✓	✓	✓	IdO's gender (female/male/undefined)
	language	✓	✓	✓	✓	preferred language (e.g. de_CH)
	urn:swissid:complies_with				✓	Possible values: EPD (elektronische Patientendossier) ZERTES (electronic signatures)
	urn:swissid:date_of_birth			✓	✓	IdO's birthday, represented as an ISO 8601:2004 YYYY-MM-DD format. Note: If the date is only partially known,

						the year can be 0000, indicating that it is omitted. the month can be 00, indicating that it is omitted. the day can be 00, indicating that it is omitted.
	urn:swissid:identity_updated_at			✓	✓	Date the document was issued by the public officer. When the document does not an issuing date (e.g. Portuguese ID Card), urn:swissid:identity_updated_at claim is populated with the date when the identity verification occurred.
	urn:swissid:place_of_birth				✓	Place of the IdO's birth according to an official identification document. This claim is not applicable for a Swiss citizen.
	urn:swissid:nationality				✓	verified nationality Iso code e.g. CHE (ISO 3166-1 alpha-3 format). If asked within the scope identity_documents, it will be returned only if the copy of the id document is available.
	urn:swissid:identity_valid_until				✓	Expiry date of the ID document used in the IdO registration process.
	urn:swissid:identity_document_type				✓	Identity document type (returned within the scope *identity_documents*). Possible values: • `passport` • `id`

	urn:swissid:identity_number				✓	Number of the identity document
	urn:swissid:age_over			✓	✓	Verified age over 16 or 18
email	email	✓	✓	✓	✓	IdO's email
phone	phone_number	✓	✓	✓	✓	IdO's phone number
address	address	✓	✓	✓	✓	IdO's **verified** address. SwissID verifies if the address it is registered as the postal address of the IdO. The address scope retrieves a JSON object containing: formatted, street_address, locality, postal_code, street_name, house_number, country, verified_at.
identity_documents	urn:swissid:identity_document_type				✓	Identity document type (returned within the scope *identity_documents*). Possible values: • `passport` • `id`
	urn:swissid:document_country				✓	Issuing country code (ISO 3166-1 alpha-3 format) of the identity document. Returned within the scope identity_documents only if the copy of the id document is available.
	urn:swissid:nationality				✓	verified nationality Iso code e.g. CHE (ISO 3166-1 alpha-3 format). If asked within the scope identity_documents, it will be returned only if the copy of the id document is available.
photoid_selfie				✓	✓	Required for downloading selfie and portrait images of the verification process
Individual claims	urn:swissid:first_name			✓	✓	All IdO's verified given names.

	urn:swissid:date_of_birth			✓	✓	IdO's birthday, represented as an ISO 8601:2004 YYYY-MM-DD format. Note: If the date is only partially known, the year can be 0000, indicating that it is omitted. the month can be 00, indicating that it is omitted. the day can be 00, indicating that it is omitted.
	urn:swissid:place_of_birth				✓	Place of the IdO's birth according to an official identification document. This claim is not applicable for a Swiss citizen.
	urn:swissid:nationality				✓	Verified nationality
	urn:swissid:age_over			✓	✓	Verified age over 16 or 18
	urn:swissid:qor		✓	✓	✓	Quality of registration process, e.g., qor2 (physical presence)
	urn:swissid:identity_updated_at			✓	✓	Date the document was issued by the public officer. When the document does not an issuing date (e.g. Portuguese ID Card), urn:swissid:identity_updated_at claim is populated with the date when the identity verification occured.
	urn:swissid:suisseid_number		✓	✓	✓	SuisseID number *(deprecated)*
	urn:swissid:identity_valid_until				✓	Expiry date of the ID document used in the IdO registration process.
	urn:swissid:place_of_origin				✓	Place of origin required for Swiss documents. String value in other (non-Swiss) documents place_of_birth

	amr	✓	✓	✓	✓	An array with the used authentication means: • urn:swissid:pwd.pwd • urn:swissid:otp.app.1fa • urn:swissid:otp.app.2fa • urn:swissid:otp.sms • pin • hwk

Tokens

The following table provides more details on OIDC and OAUTH tokens and how the RPs should manage them.

Token	Type	TTL	Revokable	Refreshable	Validation	Storable	Usage
Authorization code	Random string	2 min	no	no	no	no	For obtaining the access_token, the id_token and refresh_token
Access token	Bearer string	60 min	no	yes	yes	no	for calling userinfo
Refresh token	Bearer string	6 months	yes	yes	no	yes	for obtaining new valid refresh_token, access_token and id_token
ID token	JWT	60 min	no	yes	yes	no	Proof of IdO's authentication. Expired ID tokens should never be accepted for processing

IdO Registration flow

The RP has the option of redirecting the IdO directly to the registration page. If the goto parameter is properly defined with a valid authorization URI (see Building the authorization URI), the RP can chain the registration step with the authorization grant flow in just one call. After registration, the IdP calls the `goto` parameter redirecting the IdO to the URI defined in `redirect_uri` query string. E.g.:

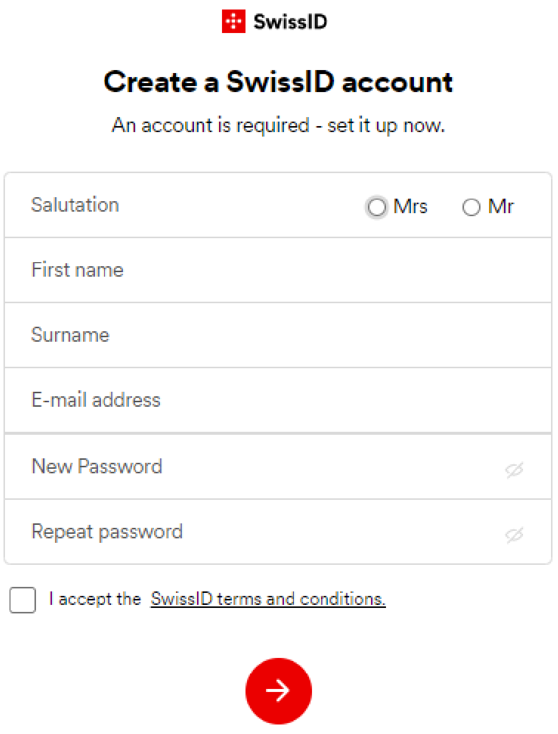
```
<ENV>/login/registration?locale=<de|fr|it|en>&goto=<AUTHORIZATION_URI>
```

The <AUTHORIZATION_URI> must be an **URL encoding** of the first call of the authorization grant flow (see Authorize endpoint), e.g.:

```
goto=<ENV>%2Fidp%2Foauth2%2Fauthorize%3Fresponse_type%3Dcode%26client_id%3D<CLIENT_ID>%26redirec
t_uri%3D<RP_CALLBACK_URL>%26scope%3Dopenid
```

Please note that <ENV>, <CLIENT_ID> and <RP_CALLBACK_URL> variables also need to be URL encoded for this case. Therefore, the `goto` parameter must be a valid call to the authorize endpoint, this happens to enable the user to grant consent before the callback to the RP happens.

Calling the registration URL will prompt this UI:


SwissID

Create a SwissID account

An account is required - set it up now.

Salutation	○ Mrs ○ Mr
First name	
Surname	
E-mail address	
New Password	
Repeat password	

☐ I accept the SwissID terms and conditions.

The IdO's registers, the IdO is presented with SwissID's consent page and finally is redirected back to the redirect_uri within the <AUTHORIZATION_URI> defined at the `goto` parameter.

IdO LoT step-up

Some RPs have business flows that require higher levels of trust (LoT) when conducted. However, the target IdO can have a lower LoT than required by the RP to accept business. Hence, SwissID offers the possibility for the RP to guide the IdO for performing a LoT step-up.

The same URL can be used to request multiple quality of registrations (i.e., LoTs), by defining the **purpose** of the step-up. Below are listed the possible purposes:

Purpose	**Outcome**
QOR1	A LoT1 identity
QOR2	A LoT2 identity
ZERTES	A LoT2 identity compliant with ZertES certification
SIGNING	A LoT2 identity compliant with ZertES certification and with SwissID Sign activated
EPD	A LoT2 identity compliant with ZertES and EPD certification and with SwissID Sign activated

Please note that although the RP requests a purpose centered in Quality of Registration (QoR), the step-up flow will always ensure that the user performs a second-factor authentication (i.e., QoA2). Furthermore, the step-up process is an encapsulation of the OpenID connect authorization flow, therefore, when the user is redirected back to the RP, it will result in a fresh authentication with new authorization code for the OpenID token exchange.

The step-up request URL follows a regular SwissID OpenID connect request, yet with two differences: a distinct URL and a new parameter purpose. For instance, requesting an account compliant with ZertES certification capable of Signing documents:

```
<a.ENV>/idcheck/rp/stepup/multi-
stepup?client_id=<CLIENT_ID>&scope=<SCOPES>&state=<STATE>&redirect_uri=<RP_CALLBACK_URL_ENCODED>
&acr_values=qoa2&claims=%7B%22userinfo%22%3A%7B%22urn%3Aswissid%3Aqor%22%3A%7B%22value%22%3A%22q
or2%22%7D%7D%2C%20%22urn%3Aswissid%3Acomplies_with%22%3Anull%7D&purpose=SIGNING
```

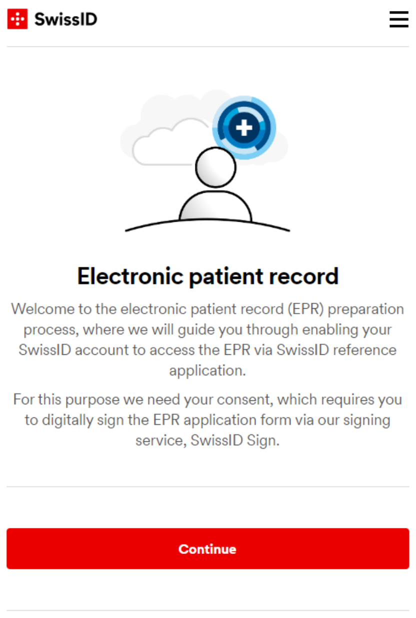
If the step-up process is successfully completed, the IdO will be automatically redirected to perform a new SwissID authentication request including the query parameters specified in the step-up URL (scope, claims, nonce, state, acr_values and redirection URI, etc). As mentioned before, for the RP it will be like the usual authentication process: the IdO is redirected to the URI specified in the *redirect_uri* parameter. If successful, the RP shall confirm the user's QoR in the resulting id_token.
Because the claims attribute held the URL encoded value of:

```
claims={"userinfo":{"urn:swissid:qor":{"value":"qor2"}}, "urn:swissid:complies_with":null}
```

After the call, an authorization code will be delivered, that in its turn can be exchanged by an id_token containing the claim:

```
"urn:swissid:complies_with": ["ZERTES"]
```

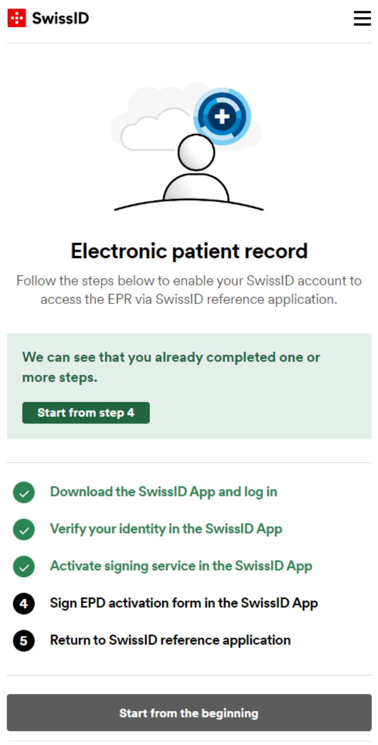
For any purpose, the IdO will be redirected to a SwissID web application which will guide the user on how to step-up its identity level. In the following figure is displayed an example UX/UI, when the RP sets the purpose to EPD:



Electronic patient record

Welcome to the electronic patient record (EPR) preparation process, where we will guide you through enabling your SwissID account to access the EPR via SwissID reference application.

For this purpose we need your consent, which requires you to digitally sign the EPR application form via our signing service, SwissID Sign.



Electronic patient record

Follow the steps below to enable your SwissID account to access the EPR via SwissID reference application.

We can see that you already completed one or more steps.

- ✓ Download the SwissID App and log in
- ✓ Verify your identity in the SwissID App
- ✓ Activate signing service in the SwissID App
- 4 Sign EPD activation form in the SwissID App
- 5 Return to SwissID reference application

Once again, when the user is redirected back, the RP can be sure that the resulting identity is of the requested level: EPD.

The following purposes are

QOR1	Identity verification process
QOR2	Certified identity verification process
ZERTES	Certified identity verification process
SIGN	Certified identity verification process and Activated Signing Service
EPD	Certified identity verification process and Activated Signing Service and activation of SwissID as an EPD compliant identifier

Additionally for some purposes a web based process is available ("[…]purpose=EPD&flow=web). If the app shall be used, this can be enforced with "[…]purpose=ZERTES&flow=app". For this purpose's a flow can be defined. If no flow is defined the standard flow defined by SwissID is used.

Web and app flows are momentarily available for the following purposes:

Purpose	Web flow available	App flow available

QOR1	Yes	Yes
QOR2	no	Yes
ZERTES	No	Yes
SIGN	No	Yes
EPD	Yes	Yes

<a.ENV>/idcheck/rp/stepup/multi-
stepup?client_id=<CLIENT_ID>&scope=<SCOPES>&state=<STATE>&redirect_uri=<RP_CALLBACK_URL_ENCODE
D>
&acr_values=qoa2&claims=%7B%22userinfo%22%3A%7B%22urn%3Aswissid%3Aqor%22%3A%7B%22value%22%3
A%22qor2%22%7D%7D%2C%20%22urn%3Aswissid%3Acomplies_with%22%3Anull%7D&**purpose=EPD&flow=web**

Finally, in case of errors during the step-up process, the user is redirected to the RP redirection URI with two additional parameters explaining the error occurred, e.g.:

```
<REDIRECT_URI>?error=no_user_consent&error_description=...
```

The following table lists the possible errors code and descriptions.

Error	**error_description**
cancelled_by_user	Operation canceled by user
general_error	<dynamic content>
invalid_client_id	invalid client
manual_check_needed	Process should manually verified
redirect_uri_mismatch	The redirection URI provided does not match a pre-defined value
no_user_consent	User did not consent operation

ID Document Service

Some business use cases involve the download of IdO's identity document pictures (e.g., passport). SwissID provides a simple service for downloading pictures used upon the identity verification of the IdO (qor1 or qor2).

To Retrieve document copy (front/back or both pictures):

```
curl GET '<a.ENV>/api-idr/public/document?side=<SIDE>' -H 'Authorization: Bearer <ACCESS_TOKEN>'
```

Please note that the ingress on `<a.ENV>` is `account` and not `login`.

- `SIDE`: front or back, if side is omitted both sides will be retrieved in the same response. Could be next values: “front”,”back”
- `ACCESS_TOKEN`: Access token bound to the IdO.

Please note also, for this end point identity_documets scope must be enabled for the RP and included into IDO Access Token.

To retrieve user's picture from document:

```
curl --location '<a.ENV>/api-idr/public/picture?depiction=portrait' \
--header 'Authorization: Bearer <IDO_ACCESS_TOKEN'
```

Please note that the ingress on `<a.ENV>` is `account` and not `login`.

- `ACCESS_TOKEN`: Access token bound to the IdO.

for this end point identity_documets scope must be enabled for the RP and included into IDO Access Token.

To retrieve user's selfie photo:

```
curl --location '<a.ENV>/api-idr/public/picture?depiction=selfie' \
--header 'Authorization: Bearer <IDO_ACCESS_TOKEN'
```

Please note that the ingress on `<a.ENV>` is `account` and not `login`.

- `ACCESS_TOKEN`: Access token bound to the IdO.

for this end point photo_id_selfie scope must be enabled for the RP and included into IDO Access Token.

Verified Postal Address

It is possible to get IdO's verified address. If the IdO fulfilled her/his postal address to the account at https://account.swissid.ch/, SwissID will verify if that name/address combination it is used for delivering postal mail. To request the verified postal, the scope address must be requested:

```
<ENV>/idp/oauth2/authorize?response_type=code&client_id=<CLIENT_ID>&redirect_uri=<RP_CALLBACK_UR
L_ENCODED>&scope=openid+address
```

When the address scope it is requested, it will be delivered in the following call to the Token endpoint within the id_token. The decoded format of the address scope is as follows, e.g.:

```
"address": {
   "locality": "Biel/Bienne",
   "postal_code": "2503",
   "country": "CH",
   "region": "BE",
   "street_name": "Zukunftstrasse",
   "house_number": "49",
   "verified_at": "2021-12-01T15:07:43.291Z",
   "qstat": 1,
   "total_score": 100,
   "street_address": "Zukunftstrasse 49",
   "formatted": "Zukunftstrasse 49\n2503 Biel/Bienne\nCH"
}
```

Besides the OIDC claims, the address scope retrieves qstat, total_score and verified_at for interpreting the address verification results.

Interpret address result

There are three attributes in the address result that must be analyzed by the RP for considering if the address verification is enough or not for the RP's address use case. The three fields to consider are qstat, total_score and verified_at:

qstat	It is used to make a statement about the quality of an address. See QSTAT for every possible value.
total_score	Match probability of the check against the reference data. We advise to never consider addresses verified with total_score below 93%.
verified_at	Checks address verification freshness. Every time the address scope it is requested, a new address verification is conducted. However, the RP should confirm that the verified_at is defined. If it is not defined (null), it means it was not possible to confirm that the IdO's address and it should be considered a self-declared address. When this happens, total_score and qstat are 0.

QSTAT

Code	Designation	Description
1	IdO hit	
2	Household hit	This household or last name (regardless of the first name) is known at this address and items can be delivered.
4	Relocation hit	A movers' address is available for this person or company.
26	International relocation hit	An unverified international address is available for this IdO.
27	Relocated, unknown	This IdO is no longer located at this address. The new address is unknown or is not permitted to be disclosed.
0	Self-declared	This address was **not** verified.

Verified Address Step-up

If the RP concludes that the retrieved address it is not valid or without enough score, it should redirect the IdO to the Address step-up flow, by calling:

```
<a.ENV>/selfmanagement/rp/stepup/address?client_id=<CLIENT>&scope=address
openid&state=<STATE>&nonce=<NONCE>&ui_locales=<en|de|fr|it>&redirect_uri=<RP_CALLBACK_URL_ENCODE
D>
```

After the address step-up, the new id_token will contain the updated address potentially verified. Nevertheless, the qstat, total_score and verified_at must be once again check, to confirm that the step-up was successful and the new address is verified.

Other useful endpoints

End session endpoint

The RPs can terminate the IdO's session (logout) by calling the following endpoint:

```
curl GET
"<ENV>/idp/oauth2/connect/endSession?id_token_hint=<ID_TOKEN>&redirect_uri=<RP_CALLBACK_URL_ENCO
DED>" -H 'Authorization: Basic <BASE64(CLIENT_ID:CLIENT_SECRET)>'
```

Revocation endpoint

Tokens issued by SwissID can be revoked by calling the following endpoint:

```
curl -X POST <ENV>/idp/oauth2/token/revoke --data "token=<TOKEN_TO_REVOKE>" --data
"client_id=<CLIENT_ID>" --data "client_secret=<CLIENT_SECRET>"
```

Introspection endpoint

The introspection endpoint enables the RP to learn real-time information about the token.

```
curl -X POST '<ENV>/idp/oauth2/introspect' -H 'Authorization: Basic
<BASE64(CLIENT_ID:CLIENT_SECRET)>'
-H 'Content-Type: application/x-www-form-urlencoded; charset=UTF-8'
--data-raw 'token=<ACCESS_TOKEN>'
```

which will retrieve a response like:

```
{
       "active": true,
       "scope": "phone openid profile email",
       "client_id": <CLIENT_ID>,
       "token_type": "Bearer",
       "exp": 1612364987,
       "sub": " s9rDnigjO/45qHH2f3t/1D1je5aL6V9uYHZctTlQiGw=",
       "iss": "https://login.swissid.ch/idp/oauth2"
}
```

BEST PRACTICES

Token and response validation

According to the OIDC specification RPs must ensure that the received tokens are valid. The following table summarizes the RP's obligations when consuming OIDC/OAUTH tokens and responses.

Step	Requirement
Authentication request	Provide state and nonce values with sufficient entropy. More on http://openid.net/specs/openid-connect-core-1_0.html#AuthRequest, http://openid.net/specs/openid-connect-core-1_0.html#NonceNotes
Validate authentication responses	Handle the state parameter correctly. More on http://openid.net/specs/openid-connect-core-1_0.html#AuthResponseValidation
Validate token endpoint responses	Validate the ID Token and proof scopes More on http://openid.net/specs/openid-connect-core-1_0.html#TokenResponseValidation
Validate ID token	RPs must validate ID Tokens as follows: 1. If the ID Token is encrypted, decrypt it using the keys and algorithms that the Client specified during Registration that the IdP was to use to encrypt the ID Token. If encryption was negotiated with the IdP at the time of Registration and the ID Token is not encrypted, the RP SHOULD reject it. 2. The Issuer Identifier for the OpenID Provider (which is typically obtained during Discovery) MUST exactly match the value of the iss (issuer) Claim. 3. The Client MUST validate that the aud (audience) Claim contains its client_id value registered at the Issuer identified by the iss (issuer) Claim as an audience. The aud (audience) Claim MAY contain an array with more than one element. The ID Token MUST be rejected if the ID Token does not list the Client as a valid audience, or if it contains additional audiences not trusted by the Client. 4. If the ID Token contains multiple audiences, the Client SHOULD verify that an azp Claim is present. 5. If an azp (authorized party) Claim is present, the Client SHOULD verify that its client_id is the Claim Value. 6. If the ID Token is received via direct communication between the Client and the Token Endpoint (which it is in this flow), the TLS server validation MAY be used to validate the issuer in place of checking the token signature. The Client MUST validate the signature of

	all other ID Tokens according to JWS [JWS] using the algorithm specified in the JWT alg Header Parameter. The Client MUST use the keys provided by the Issuer. 7. The alg value SHOULD be the default of RS256 or the algorithm sent by the Client in the id_token_signed_response_alg parameter during Registration. 8. If the JWT alg Header Parameter uses a MAC based algorithm such as HS256, HS384, or HS512, the octets of the UTF-8 representation of the client_secret corresponding to the client_id contained in the aud (audience) Claim are used as the key to validate the signature. For MAC based algorithms, the behavior is unspecified if the aud is multi-valued or if an azp value is present that is different than the aud value. 9. The current time MUST be before the time represented by the exp Claim. 10. The iat Claim can be used to reject tokens that were issued too far away from the current time, limiting the amount of time that nonces need to be stored to prevent attacks. The acceptable range is Client specific. 11. If a nonce value was sent in the Authentication Request, a nonce Claim MUST be present, and its value checked to verify that it is the same value as the one that was sent in the Authentication Request. The Client SHOULD check the nonce value for replay attacks. The precise method for detecting replay attacks is Client specific. 12. If the acr Claim was requested, the Client SHOULD check that the asserted Claim Value is appropriate. The meaning and processing of acr Claim Values is out of scope for this specification. 13. The auth_time Claim. The Client SHOULD check the auth_time Claim Value and take appropriate action if it determines too much time has elapsed since the last End-User Authentication. Please note that auth_time represents the time after the authentication took place and not the time the user gave consent for the requested scopes. Therefore, the auth_time represents the time just before the consent screen is displayed to the End-User. Time of the SwissID IdP is synchronized and returns UTC. More on http://openid.net/specs/openid-connect-core-1_0.html#IDTokenValidation
Protect Client ID and secret	RP’s credentials (i.e., client secret) must be stored safely for remaining a secret only known by the RP. RPs should inform the SwissID IdP in case credentials have been compromised.
Store tokens securely	Tokens, especially Refresh Tokens, must be treated as credentials and stored securely in a place where only the End-Users for whom they were issued can access them.
Follow SLA	SLA, in term of functional and nonfunctional requirements must be considered.

Cryptographic key rotation

As a central security element, the SwissID uses cryptographic keys for the signing of the communication with the RP. The lifetime of these keys is limited and therefore the keys will have to be renewed on a regular basis. This causes the public keys in the jwks_uri endpoint to change. Since the RPs always have to validate the JWTs (e.g., the ID Token) sent by SwissID it is mandatory that the RPs always use the correct public key for this. The jwks_uri endpoint contains several public keys that can be uniquely identified with so-called key identifiers (kid). In the header of the decoded JWT the key identifier is contained, with which the public key (for validation) can be identified in the jwks_uri endpoint. As far as this is considered, the renewal of the public key with the certificate renewal will not have negative consequences for the RPs. More on https://openid.net/specs/openid-connect-core-1_0.html#RotateSigKeys

Single Sign-On

"***In OpenID Connect, the session at the RP typically starts when the RP validates the End-User's ID Token.***"
in OIDC Standard - Session Management.

The Single Sign-on (SSO) concept happens when an IdO logs-in to an application and is then automatically signed into other applications. For instance, if a user logs into Gmail, (s)he is automatically authenticated into YouTube.

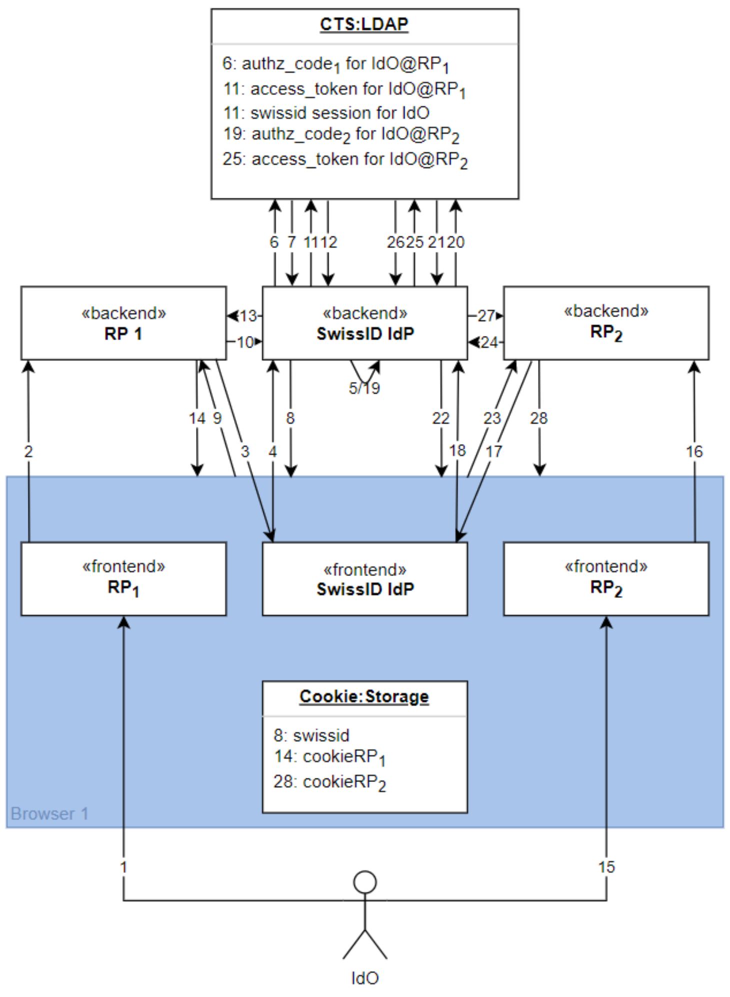
Session Cookies

Typically, SSO schemes rely on HTTP cookies (known as session cookies in this context). A HTTP cookie (browser cookie) is a small piece of data that a server sends to the user's web browser. The browser may store it and send it back with later requests to the same server. This is used to tell if two requests came from the same browser — keeping a IdO logged-in, for example. I.e., it is a way, for enabling statefulness in the stateless HTTP protocol. Cookies are used for other purposes than session management, nevertheless, we will only overview session cookies.

After receiving an HTTP request, a server can send one or more Set-Cookie headers with the response. The cookie is usually stored by the browser, and then the cookie is sent with requests made to the same server inside a Cookie HTTP header. An expiration date or duration can be specified, after which the cookie is no longer sent. Additional restrictions to a specific domain and path can be set, limiting where the cookie is sent.

SwissID SSO

SwissID follows the same concept: if there is a valid session cookie in the browser, the authentication step it is jumped and it is immediately redirected (with the new authorization code) to the calling RP domain. Please, see the below diagrams for more details.



Steps:

0. (before) Cookie:Storage and CTS:LDAP are empty.
1. IdO browses doman1.com from RP_1.
2. IdO accesses some protected resource at RP_1.
3. RP_1 redirects the IdO to SwissID IdP.
4. IdO is redirected.
5. IdO authenticates at SwissID IdP.
6. SwissID IdP creates OIDC authorization code ($authzCode_1$) for $IdO@RP_1$.
7. Stores $authzCode_1$ in CTS.
8. Browser receives and stores the SwissID session cookie: $swissid_1$.
9. Browser redirects $authzCode_1$ to RP_1.
10. RP_1 starts exchanging $authzCode_1$ for the OIDC tokens.
11. SwissID IdP checks if the $authzCode_1$ exists.
12. SwissID gathers the respective OIDC tokens from CTS.
13. RP_1 receives OIDC tokens and the exchange ends.
14. Browser stores RP_1 session cookie.
15. In the same browser, IdO accesses domain2.com from RP_2.

16. IdO accesses some protected resource at RP_2.
17. RP_2 redirects user to SwissID IdP.
18. IdO is redirected.
19. If $swissid_1$ session cookie is still valid, jump authentication step - note that no new session is created. Otherwise, the IdO must re-authenticate.
20. SwissID IdP creates $authzCode_2$ for IdO@RP_2
21. And stores it in CTS.
22. Browser receives $authzCode_2$ to RP_2.
23. Browser forwards $authzCode_2$ to RP_2.
24. RP_2 starts exchanging $authzCode_2$ for the OIDC tokens.
25. SwissID IdP checks if the $authzCode_2$ exists.
26. SwissID gathers the respective OIDC tokens from CTS.
27. RP_2 receives OIDC tokens and the exchange ends.
28. Browser stores RP_2 session cookie.

Remember Me

OIDC does not define how to achieve the “Remember Me” behavior yet. When the RP redirects the IdO to the SwissID, the authentication step(s) may be jumped if the calling device holds a valid SwissID session cookie. However, SwissID session time-to-live (TTL) could be different from the RP requirements. In this case, the RP should complement SwissID SSO cookie with its own Remember Me strategy, for instance by issuing its own session cookies alongside SwissID SSO cookie.