Technical information: Integrating swiss-rx-login into your website

Using swiss-rx-login, medical personnel in Switzerland are able to access the secure areas of websites of participating Swiss health companies and organisations by using their personal single sign-on offered free of charge by the Refdata foundation.

The technical integration into your website is simple and does not require a great deal of work in terms of software development, as we support two integration variants: one a light proprietary protocol (legacy), the other the standard OAuth2 authentication. We suggest using OAuth2.

In the proprietary protocol, available since early 2010, authentication is carried out by doing a Client Redirect from the company website to https://swiss-rx-login.ch and passing identity parameters such as company GLN and PostBackURL via POST or GET. Refdata then presents the end user with a registration dialogue (or verifies an already existing registration). Once authentication has been successful, the client is redirected from swiss-rx-login back to your company website "PostBackURL" (which must be registered on swiss-rx-login), passing a set of parameters such as access type and control hash.

In the OAuth2 variant, available since late 2017, authentication is carried out in a very similar fashion as in the legacy proprietary protocol, but based on the OAuth 2.0 standard as definied in IETF RFC 6749.

Version 2017-10-01

What is the easiest way to start using the service?

1. If your company has not yet registered service admins with us, go to www.refdata.ch and read the information under "Für Firmen > Anmeldung" on how to proceed.
2. Then define which two persons are responsible for managing swiss-rx-login in your company (registering postback-URLs, managing users), fill out the request form (also available from refdata.ch) and send it to us. We will create these two users on swiss-rx-login and give them "ServiceAdmin" permissions so that they can access the necessary functions: For reasons of security, all PostBackURLs must be registered with **swiss-rx-login.ch**. In addition, each client has its own individual id and secret (Client ID, Client Secret). The secret is a code as a shared key.
3. Create your content website using cookie- or session-based authentication.
4. Put an html form with a login/authentication button on your (unprotected) startpage. Clicking this button shall then launch the authorization process, either based on the legacy proprietary protocol or using OAuth 2.0

Using OAuth 2.0

Swiss-Rx-Login implements OAuth 2.0 according to IETF RFC 6749. However, only the two main grant types are available:

- Authorization Code: used with server-side applications where you can keep your secret protected. For your typical company website.
- Implicit: used with standalone Mobile Apps or pure javascript Web Applications, where it is not possible to protect the Client Secret.

The exact documentation of the OAuth2.0 protocol is available in the RFC mentioned above or in various tutorials online or the inofficial website.

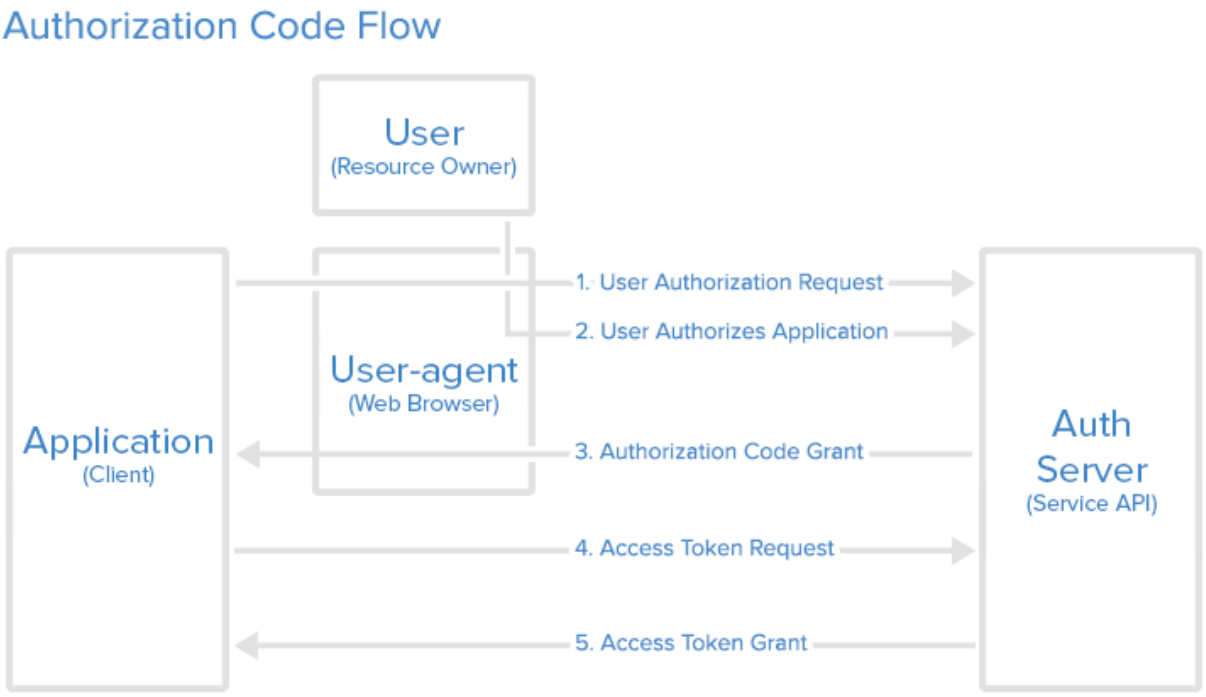
The most important parameters for Swiss-Rx-Login are:

- The authorization endpoint URL is https://swiss-rx-login.ch/oauth/authorize
- The token endpoint URL is https://swiss-rx-login.ch/oauth/token
- The allowed scope values are anonymous and personal (includes some details)

The OAuth2 flows and parameters

The simplified flow diagrams for the two grants look as follows. Both use a quite similar set of parameters. All HTTP request must use SSL!

Authorization Code grant:



1. *myWebSite through user-agent web browser to swiss-rx-login*
   https://swiss-rx-login.ch/oauth/authorize?
   response_type=authorization_code&client_id=CLIENT_ID&
   redirect_uri=CALLBACK_URL&scope=anonymous&state=randomStateString

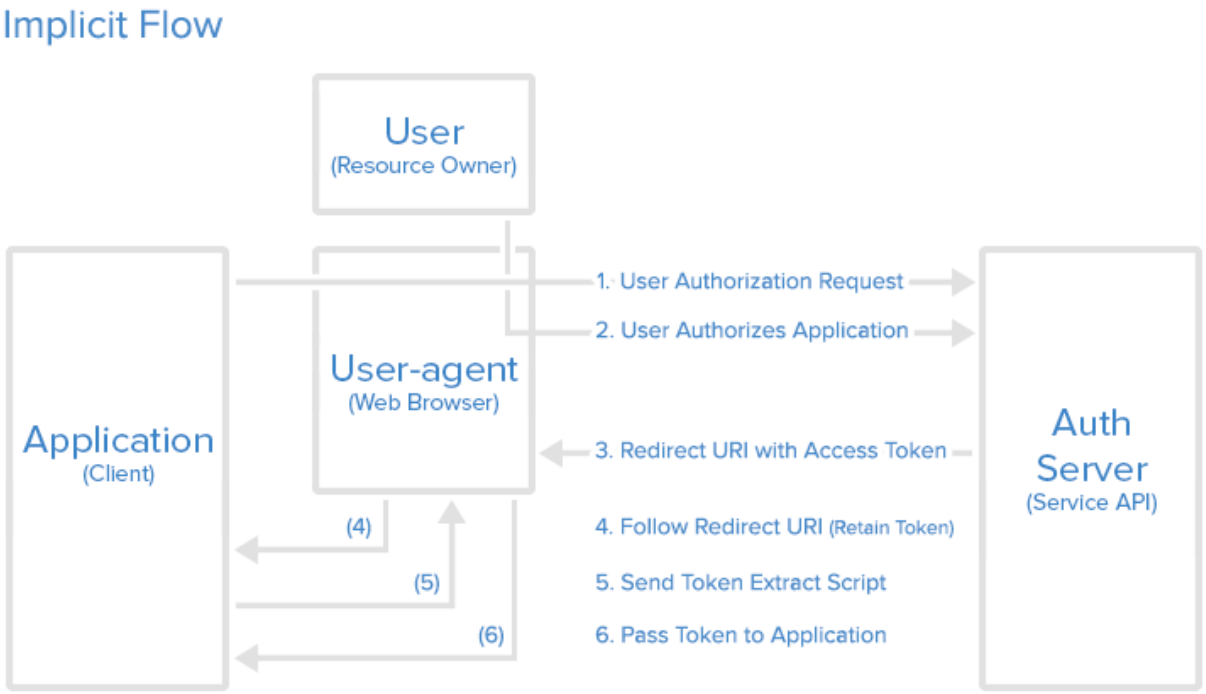
2. *User Logs in*

3. *Swiss-rx-login to myWebSite*
   https://myWebSite.com/callback?code=AUTHORIZATION_CODE

4. *myWebSite to swiss-rx-login*
   https://swiss-rx-login.ch /oauth/token?client_id=CLIENT_ID&
   client_secret=CLIENT_SECRET&grant_type=authorization_code&
   code=AUTHORIZATION_CODE&redirect_uri=CALLBACK_URL

5. *swiss-rx-login to myWebSite*
   https://myWebSite.com/callback?token=JWT-Token

Implicit grant:



1. *myWebSite to swiss-rx-login*
   https://swiss-rx-login.ch/oauth/authorize?
   response_type=token&client_id=CLIENT_ID&
   redirect_uri=CALLBACK_URL&scope=anonymous

2. *User Logs in*

3. *Swiss-rx-login to myWebSite*
   https://myWebSite.com/callback?token=JWT-Token

An explanation of the detailed parameters used is available on the next page.

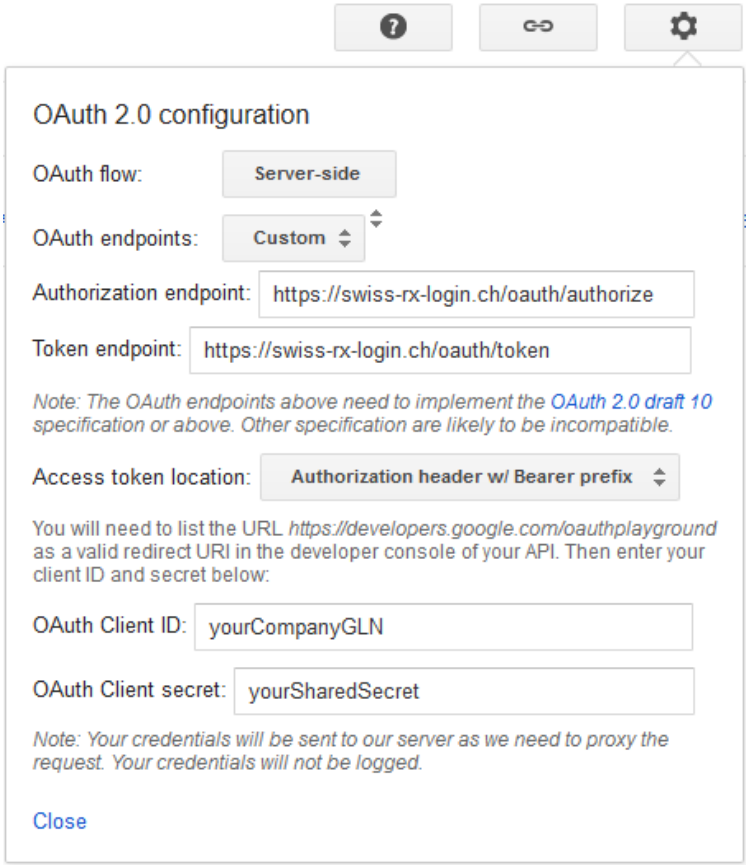
Step	Name	Legacy*	Description	Example	
1	response_type		"token" or "authorization_code"	token	Req
1 + 4	client_id	GLN	Your client id from swiss-rx-login (GLN)	7601001234567	Req
1 + 4	redirect_uri	BackURL	URI where you want to receive the authorization code (1) or the token (4)	https://myWebsite.com/callback	Req
1	scope	Identity	The extent of identity info needed by your website to allow the login.	anonymous personal	Opt
1, 3	state		A state object from the client in string format, e.g. base64 json, will be posted back, useful to eliminate XSRF or handle session state.	eyAicGFyYW0xIiA6ICJhYmMiICwgInBhcmFtMiIgOiAzNDUgfQ==	Opt
3 + 4	code		Authorization code returned from swiss-rx-login after the user has logged in successfully	kjekgf9k4hgofy88fvn3==	Req
4	client_secret		Your client secret from swiss-rx-login	hkvjkjr8ösdfigh8srd7843hufh8=	Req
4	grant_type		Always the constant "authorization_code"	authorization_code	Req
5	token		JWT Token	eyJhbGci...	Req
1	lang	Lang	Language of the login dialog. (Fallback: according to browser settings) Allowed: DE/FR/EN	DE	Opt
1 + 4	showtext	ShowText	Information string shown to the user after login	myWebSite	Opt
1	types	Types	A string that contains all the AccTypes accepted for entry into your website	A	Opt

**The legacy name is the name of the identical/similar parameter in the proprietary legacy protocol of swiss-rx-login, the way it was available since 2010. It is very similar to OAuth 2.0.*

Req = required
Opt = optional

Testing it in the Google OAuth 2.0 Playground (Authorization Grant, Server-side OAuth flow)
Google offers a nice online playground to experiment with any OAuth2 implementation:

- Make sure that in swiss-rx-login, you did register the Google example URL (https://developers.google.com/oauthplayground), using your ServiceAdmin login.
- In addition, know your client ID (the company GLN, to use as client ID) and the client secret (Shared Secret), but visible from your company page in Swiss-Rx-Login.
- Configure the playground for authorization grant, using the dropdown at the top right:



- Once this is done, you can simply switch over to the left side of the playground, select "Step 1" and enter the desired scope:
  - Anonymous (if you only want to know if the user has a valid account)
  - Personal (if you want to get some details about the user)
- Now click on [Authorize APIs] to finish Step 1. The playground will request an authorization token from Swiss-Rx-Login by going to https://swiss-rx-login.ch/oauth/authorize?scope=personal&redirect_uri=https://developers.google.com/oauthplayground&response_type=code&client_id=YourCompanyGLN
- Once the login in SRXL is successful, the client will return to the Google Playground, providing the authorization code (e.g. 902e4f37-7030-4266-b49e-47a9c93fa9c6c00343cb-64f7-4eab-8a47-943678395bc3) attached to the postback URL, in a GET HTTP request inside the user-agent/browser (https://developers.google.com/oauthplayground/?code=b8401e67-c35e-483c-8260-a146a9eeda1babb74a1b-2e8d-4a36-b85a-b80a093b758f)

- Now you can simply click [Exchange authorization code for tokens] to do Step 2. The playground will take the authorization code obtained in Step 1 and exchange it with SRXL (outside the user-agent, server-to-server) for an Access token through HTTP POST, using the client ID and client secret, e.g. https://swiss-rx-login.ch/oauth/token?code=b8401e67-c35e-483c-8260-a146a9eeda1babb74a1b-2e8d-4a36-b85a-b80a093b758f&redirect_uri=https%3A%2F%2Fdevelopers. google.com %2Foauthplayground&client_id=yourClientID& client_secret=yourSharedSecret&scope=&grant_type=authorization_code)
- The response to the above request will be a JSON object, a wrapper around the token itself:

{
"access_token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJyb2xlIjoic3dpc3NSeExvZ2luIiwibmFtZWlkIjoiSXJ1OUdFS3BEYWp4Zm5xdkNLZTZoQT09MDAwMDAwMDAiLCJodHRwczovL3N3aXNzLXJ4LWxvZ2luLmNoL29hdXRoL2NsYWltcy9BY2NJRCI6IklydTlHRUtwRGFqeGZucXZDS2U2aEE9PTAwMDAwMDAwIiwiaHR0cHM6Ly9zd2lzcy1yeC1sb2dpbi5jaC9vYXV0aC9jbGFpbXMvQWNjVHlwZSI6IkEiLCJodHRwczovL3N3aXNzLXJ4LWxvZ2luLmNoL29hdXRoL2NsYWltcy9BY2NHcnAiOiJQSEFSTSIsImdpdmVuX25hbWUiOiJUaG9tYXMiLCJmYW1pbHlfbmFtZSI6IlfDpGx0aSIsImVtYWlsIjoidGhvbWFzLndhZWx0aUBlLW1lZGlhdC5uZXQiLCJodHRwOi8vc2NoZW1hcy54bWxzb2FwLm9yZy93cy8yMDA1LzA1L2lkZW50aXR5L2NsYWltcy9zdHJlZXRhZGRyZXNzIjoiMzAyNyBCZXJuIiwibGFuZ3VhZ2UiOiJERSIsImdsbiI6Ijc2MDEwMDMxNzg5OTkiLCJ1bmlxdWVfbmFtZSI6IlRob21hcyBXw6RsdGkiLCJuYmYiOjE1MDY2MTQ4MzUsImV4cCI6MTUwNjYxODQzNSwiaWF0IjoxNTA2NjE0ODM1LCJpc3MiOiJodHRwczovL3N3aXNzLXJ4LWxvZ2luLmNoIiwiYXVkIjoiNzYwMTAwMTM2MjM4MyJ9.eIZGOWlEtFedbNilD8QN1qQYwcq1xIy7sgkOYCFQqJQ",
"token_type": "Bearer",
"expires_in": 3600
}

The resulting token

A typical JWT result token might look like this:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJyb2xlIjoic3dpc3NSeExvZ2luIiwibmFtZWlkIjoiSXJ1OUdFS3BEYWp4Zm5xdkNLZTZoQT09MDAwMDAwMDAiLCJodHRwczovL3N3aXNzLXJ4LWxvZ2luLmNoL29hdXRoL2NsYWltcy9BY2NJRCI6IklydTlHRUtwRGFqeGZucXZDS2U2aEE9PTAwMDAwMDAwIiwiaHR0cHM6Ly9zd2lzcy1yeC1sb2dpbi5jaC9vYXV0aC9jbGFpbXMvQWNjVHlwZSI6IkEiLCJodHRwczovL3N3aXNzLXJ4LWxvZ2luLmNoL29hdXRoL2NsYWltcy9BY2NHcnAiOiJQSEFSTSIsImdpdmVuX25hbWUiOiJUaG9tYXMiLCJmYW1pbHlfbmFtZSI6IlfDpGx0aSIsImVtYWlsIjoidGhvbWFzLndhZWx0aUBlLW1lZGlhdC5uZXQiLCJodHRwOi8vc2NoZW1hcy54bWxzb2FwLm9yZy93cy8yMDA1LzA1L2lkZW50aXR5L2NsYWltcy9zdHJlZXRhZGRyZXNzIjoiMzAyNyBCZXJuIiwibGFuZ3VhZ2UiOiJERSIsImdsbiI6Ijc2MDEwMDMxNzg5OTkiLCJ1bmlxdWVfbmFtZSI6IlRob21hcyBXw6RsdGkiLCJuYmYiOjE1MDY2MTQwMDAsImV4cCI6MTUwNjYxNzYwMCwiaWF0IjoxNTA2NjE0MDAwLCJpc3MiOiJodHRwczovL3N3aXNzLXJ4LWxvZ2luLmNoIiwiYXVkIjoiNzYwMTAwMTM2MjM4MyJ9.0fhyHzikmn5maEETm5HvWAzK8TgQZ9VyltfxbhOCLpo

If you base64-decode this, you will get a two JSON objects that contain a header and a payload element plus the signature block signed in HS256.
To sign the token, the server used 2* the client secret (if your shared secret is “ABC123456”, the code used to sign/verify the token is “ABC123456ABC123456”); this is due to the fact that some of the original shared secrets inside Swiss-Rx-Login aren’t long enough). To verify the signature in your code, follow an online documentation such as https://auth0.com/docs/api-auth/tutorials/verify-access-token

To manually decode it and verify the signature if needed, use an online service.

Once decoded, the token containing the user information looks similar to:

Header:

{
 "typ": "JWT",
 "alg": "HS256"
}

Payload

{
 "role": "swissRxLogin",
 "nameid": "Iru9GEKpDajxfnqvCKe6hA==00000000",
 "https://swiss-rx-login.ch/oauth/claims/AccID": "Iru9GEKpDajxfnqvCKe6hA==00000000",
 "https://swiss-rx-login.ch/oauth/claims/AccType": "A",
 "https://swiss-rx-login.ch/oauth/claims/AccGrp": "PHARM",
 "given_name": "Thomas",
 "family_name": "Wälti",
 "email": "thomas.waelti@e-mediat.net",
 "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/streetaddress": "3027 Bern",
 "language": "DE",
 "gln": "7601003178999",
 "unique_name": "Thomas Wälti",
 "nbf": 1506596539,
 "exp": 1506600139,
 "iat": 1506596539,
 "iss": "https://swiss-rx-login.ch",
 "aud": "7601001049369"
}

Testing it in the Google OAuth 2.0 Playground (Implicit Grant, Client-side OAuth flow)

You can also test and learn about the other grant type using the playground. Simply change the config of the playground to use the Client-side OAuth flow in the settings. As no Shared Secret is needed, the input box will disappear.

As this is only a one-step workflow, the access token will be returned immediately.

{
  "access_token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJyb2xlIjoic3dpc3NSeExvZ2luIiwibmFtZWlkIjoiSXJ1OUdFS3BEYWp4Zm5xdkNLZTZoQT09MDAwMDAwMDAiLCJodHRwczovL3N3aXNzLXJ4LWxvZ2luLmNoL29hdXRoL2NsYWltcy9BY2NJRCI6IklydTlHRUtwRGFqeGZucXZDS2U2aEE9PTAwMDAwMDAwIiwiaHR0cHM6Ly9zd2lzcy1yeC1sb2dpbi5jaC9vYXV0aC9jbGFpbXMvQWNjVHlwZSI6IkEiLCJodHRwczovL3N3aXNzLXJ4LWxvZ2luLmNoL29hdXRoL2NsYWltcy9BY2NHcnAiOiJQSEFSTSIsImdpdmVuX25hbWUiOiJUaG9tYXMiLCJmYW1pbHlfbmFtZSI6IlfDpGx0aSIsImVtYWlsIjoidGhvbWFzLndhZWx0aUBlLW1lZGlhdC5uZXQiLCJodHRwOi8vc2NoZW1hcy54bWxzb2FwLm9yZy93cy8yMDA1LzA1L2lkZW50aXR5L2NsYWltcy9zdHJlZXRhZGRyZXNzIjoiMzAyNyBCZXJuIiwibGFuZ3VhZ2UiOiJERSIsImdsbiI6Ijc2MDEwMDMxNzg5OTkiLCJ1bmlxdWVfbmFtZSI6IlRob21hcyBXw6RsdGkiLCJuYmYiOjE1MDY2MTUyMTUsImV4cCI6MTUwNjYxODgxNSwiaWF0IjoxNTA2NjE1MjE1LCJpc3MiOiJodHRwczovL3N3aXNzLXJ4LWxvZ2luLmNoIiwiYXVkIjoiNzYwMTAwMTM2MjM4MyJ9.j1IsxxZrEzxKxq43OKXM-8lDAiI1HF4u5KNWonw_rGc",
  "token_type": "Bearer",
  "expires_in": 3600
}

Using the legacy proprietary protocol

5. Submit at least the parameters GLN (of your company) and PostBackURL (where you want us to callback once authentification is ok) to http://swiss-rx-login.ch/ (either as POST or GET).
   (Make sure that the URL passed as PostBackURL parameter has been registered by one of your ServiceAdmins in the "Company Administration" section of swiss-rx-login for your company)
6. swiss-rx-login then verifies the user, asking him to login if necessary, and then posts back to your PostBackURL. In there, parse the POST variables that you're receiving - if you get a variable "AccType" with a value of "A", the user was authenticated by us and you can then also authenticate him/her to access your site (by e.g. setting a cookie).

Parameters to be submitted to swiss-rx-login (using HTTP POST or GET)

Param	Description, value area	Example	Type
GLN	Your company's registered GS1-GLN/EAN code (NOT the GLN of the ServiceAdmin person!)	7601001001878 (visible in SRXL > "Firma verwalten")	1
BackURL	Registered (!) reply address of your server (PostbackURL, as configured in the AdminPortal)	http://www.example.com/biblio/login.aspx	1
Lang	Optional: Language of the login dialogue (fallback value: according to browser settings) Possible values: DE/FR	FR	2
ShowText	Optional: Information string, if other than the registered company name should appear in the swiss-rx-login dialogue.	"Example Infodoc" max. length 40 char.	2
Identity	Optional: The extent of identity needed by your website to allow the login. Possible values are: ANONYMOUS (default): Anonymous, you only get back the AccType of the user) PERSONAL: Including GLN, Name, Adr (City), Email (only provided if the end user has given his consent for your website to get this info)	ANONYMOUS / PERSONAL	
Types	Optional: A string that contains all the AccTypes accepted for entry into your website	A	

Parameters delivered to the PostBackURL of the company site (via POST)

Param	Description, value area	Example	Type
AccType	Access type: A=Academic medical professional (or employee for his own company website) B = Employee (Mitarbeiter im Gesundheitswesen) C = Company (Company account, used as serviceaccounts for machine-based services)	A	1
AccID	Access ID: An anonymous Identifier to identify a user on subsequent visits, to e.g. tie him to a user account on your website that he created there. Always the same for the same user (one-way hash value of the userGLN + YourCompanyGLN + salt)	39e4420ba0a27a561 477ed68b6b6a73a	2
AccGrp	List of access groups (1…n, multi entries possible, separated by comma): • MED=Doctor • PHARM=Pharmacist • DENT=Dentist • VET = Veterinary surgeon • EMP = Employee • ADM = Employee (ServiceAdmin) • COMP = Company (ServiceAccount)	MED,PHARM	2
TS	UNIX time stamp	1334921916	2
Hash	Hash value for whole parameter set based on the key agreed (SHA1, base64 encoded)	YELmJzEKx3mHmMM fzvm1F3FftpY=	2
UsrGLN	*Optional: GS1-GLN of authorised person*	*7601003006070*	*3*
UsrName	***Name according to RefData entry, without title***	***Hans Meier***	***3***
UsrAdr	*Short address according to RefData entry*	*3006 Berne*	*3*
UsrID	*Unique User ID in swiss-rx-login. Is either the GLN (if available) or the e-mail originally used when signing up for swiss-rx-login.*	*7601003006070 OR drorig@ovana.ch*	*3*
UsrLang	*User language (de/fr)*	*de*	*3*
UsrEmail	*Current E-Mail address of this user, as entered in swiss-rx-login user profile. If you use an e-mail address in your portal, please use this – this makes sure that the user keeps it current in ONE place for all pharma websites.*	*drnow@ovana.ch*	*3*

Type 1: Parameters always provided, must be evaluated.

Type 2: Parameters always provided; use by company website optional. Time stamp and hash value serve to increase data security; AccGrp for those cases in which a more sophisticated authorization hierarchy is necessary (e.g. differentiation between doctor and pharmacist).

Type 3: Parameters that permit the person to be traced;
only provided if the end user concerned has given his consent

Recommendation: Calculating the hash value to check the value received

It is also possible to check the parameters provided by **swiss-rx-login.ch** by creating a string using the parameters provided by **swiss-rx-login.ch** to the PostBackURL together with the company key and then calculating the hash value from this via SHA1 (and base64 encoding it). This must correspond with the hash value provided by **swiss-rx-login.ch**.

VB.Net sample code for the hash input: Add all parameters via "_" (underscore):

```
String.Format("{0}_{1}_{2}_{3}_{4}_{5}_{6}", AccType, AccGrp, UsrGLN, UsrName, UsrAdr, CStr(TS), sharedKey)
```

Sample data:

```
A_MED,PHARM_7601000123456___1258474630_TEST
```

Resulting SHA1 hash, base64 encoded:

```
ayr0ZmR8Ghj81ElRBgbiD5h1ZrY=
```

You can pass your own parameters, too

You can post your own data to swiss-rx-login and receive it again in the PostBack: simply attach your own variables to the PostBackURL parameter that you are sending us and you will receive it again in the postback, allowing you to get them by parsing the querystring. Here are two examples for passing parameters:

a) Using POST in a form:

```
<input type="hidden" name="BackURL" value="https://mytestsite.com/ srxl-prod.php?MyOwnSessionID=whatever" />
```

b) Using GET in a URL (do not forget about URL/HTML encoding):

```
<a href="https://swiss-rx-login.ch/?GLN=760100YOURCOMPANY&Lang=fr&ShowText=YOURCOMPANY&BackURL=https%3A%2F%2Fmytestsite.com%2Fsrxl-prod.php%3FMyOwnSessionID%3Dwhatever">swiss-rx-login</a>
```

Readout example for your PostBackURL page, code snippet in PHP:

```
echo "MyOwnSessionID: ".$_GET["MyOwnSessionID "]."<br />";
```

This additional feature has two advantages:

- To pass different query parameters (e.g. german and french sites), you do not have to register multiple sites on our ServiceAdmin site.
- You can better integrate the SRXL identification service into your own systems.

Integration tips

Make sure to use the correct GLN and that your PostBackURL is registered!

Use the GLN of your company, not a personal one! Go to the admin portal, login and make sure that your PostBackURL is registered – at the top of the page, the GLN of your company is clearly visible.

Implementation via own cookies on company website

The successful website authorization via **swiss-rx-login.ch** can be persistently stored in a cookie with the client and verified when visiting the site later; this reduces the load on the service and the nuisance for the end user. In such a case, we recommend limiting the validity of the cookie to a month; this ensures that the user authentication remains current.

Granting access to the relevant websites to internal employees

To enable access to swiss-rx-login secured websites to your own employees, we recommend that the origin IP of the user is checked first (e.g. via REMOTE_ADDR or REMOTE_HOST server variables): Authentication using swiss-rx-login is only needed for external users., and not for internal IPs. Employees belonging to authorised access groups such as doctors or pharmacists can of course register directly via their personal GLN. In addition, the account information of your Service-Administrators enables them to authenticate in swiss-rx-login (but only for their own servers). There is also the possibility to create "CompanyUsers" in our admin portal (just mail us at tech@swiss-rx-login.ch so that we can enable this feature for your company) – such users have the same access rights to your protected websites as a ServiceAdmin has. This is a good solution for product and marketing people who might need access wherever they are.

Testing the service

You can test your integration directly against our productive server. Just use your ServiceAdmin account credentials (GLN, password) on swiss-rx-login.ch (due to legal and security restrictions, this only works for websites of companies where this person is registered as ServiceAdmin).

After entry and a successful check by our server, the test user's browser is redirected by our server to the BackURL provided by you. You can now check the parameters that were sent and authorize the user's access to the secure websites.

Support and Help

If you need additional support, please contact us using the information in the footer of this document.

Proprietary legacy protocol: example code

If you have examples in other programming languages, please email them to us (tech AT swiss-rx-login.ch) and we will add them to this documentation.

HTML when calling/redirecting to swiss-rx-login

Important: Use your Company-GLN for the GLN value, and NOT your personal account GLN!

```
<form name="formPost" action="https://swiss-rx-login.ch/" method="post">
<input type="hidden" name="GLN" id="GLN" value="7601001300999"/>
<input type="hidden" name="BackURL" id="BackURL"
value="https://testpharm.rxpub.net/return.asp"/>
<input type="hidden" name="ShowText" id="ShowText" value="TestPharm"/>
<input type="hidden" name="Lang" id="Lang" value="DE"/>
<input type="submit" value="Authenticate">
</form>
```

Querying PostBack

Be aware that these are very basic examples. The most basic function would just check if the AccType POSTed from our server to your PostBackURL-Call is not null, but "A". We suggest that you at least also check if the page callback indeed comes from swiss-rx-login (if you don't want to implement the hash calculation)

In ASP.NET

```
Dim accType As String = Request.Params("AccType")
```

In "classic" ASP

```
<%
dim accType
accType=Request.Form("AccType")

If accType="A" Then
Response.Write("Access Type : " & accType)
Response.Write("<br />Login successful !")
Else
Response.Write("Login failed")
End If
%>
```

In PHP

Save this example page as a php file on your server, then change the URL in the hidden input "BackURL" to point to the saved file. As "GLN", put in your company's GLN (the example uses the e-mediat GLN). You might also want to adapt the "ShowText".

```
<html>
<body>
<h2>swiss-rx-login.ch: PHP example code snippet</h2> <h3>This page contains protected content.</h3>

<?php
//Hash test
error_reporting(E_ALL);

$ref = $_SERVER['HTTP_REFERER'];
if ($ref != "")
  echo "Referer: ".$ref."</br>";

$verifyHash = "";
if (strlen(strstr($ref,"://swiss-rx-login.ch"))>0)
{
  echo "Received Postback Parameters: "."</br>";
  echo "AccType: ".$_POST["AccType"]."</br>";
  echo "AccGrp: ".$_POST["AccGrp"]."</br>";
  echo "UsrGLN: ".$_POST["UsrGLN"]."</br>";
  echo "UsrName: ".$_POST["UsrName"]."</br>";
  echo "UsrAdr: ".$_POST["UsrAdr"]."</br>";
  echo "TS: ".$_POST["TS"]."</br>";
  echo "Hash: ".$_POST["Hash"]."</br>";
  echo "POSTMyOwnSessionIDParam: ".$_POST["POSTMyOwnSessionIDParam"]."</br>";
  echo "GETMyOwnSessionIDFormAction: ".$_GET["GETMyOwnSessionIDFormAction"]."</br>";
  echo "GETMyOwnSessionIDinBackURL: ".$_GET["GETMyOwnSessionIDinBackURL"]."</br>";
  echo "</br>"."<hr>";

  $sharedKey = "YourSecretSharedKeyAsAvailableFrom";

  $verifyStr = $_POST["AccType"] ."_". $_POST["AccGrp"] ."_". $_POST["UsrGLN"] ."_".
$_POST["UsrName"] ."_". $_POST["UsrAdr"] ."_". $_POST["TS"] ."_". $sharedKey;
  $verifyHash = base64_encode( pack( "H*", sha1( $verifyStr ) ) );

  echo "Concatenated string of PostBack parameters:</br>";
  echo $verifyStr."</br>";
  echo "SHA1 hash, base64 encoded:</br>";
  echo $verifyHash."</br>";

  echo "<hr>";
}

if ($_POST["AccType"] == "A")
```

```
{
     echo "AccType=A: You were  authenticated by swiss-rx-login as beloging to the group " .
$_POST["AccGrp"] . "</br>";
  if ($_POST["Hash"] == $verifyHash)
  {
    echo "Hash=Verified: In addition, the hash code verification was ok, nobody tampered with the
data.";
  }
  else
  {
    echo "Hash=Failed: However, the hash code verification failed, check your exchangeKey setting.";
  }
     ?>
     <p style="color:blue;">Now you can see it all.</p>
     <?php
}
?>

</body>
</html>
```

In Javascript

First, you need a function to get an array of query parameters

```
location.querystring = (function() {
// The return is a collection of key/value pairs
var queryStringDictionary = {};
// Gets the query string, starts with '?'
var querystring = decodeURI(location.search);
if (!querystring) {
    return {};
}
// Remove the '?' via substring(1) and '&' seperates key/value pairs
querystring = querystring.substring(1);
var pairs = querystring.split("&");

// Load the key/values of the return collection
for (var i = 0; i < pairs.length; i++) {
    var keyValuePair = pairs[i].split("=");
    queryStringDictionary[keyValuePair[0]] = keyValuePair[1];
}
// toString() returns the key/value pairs concatenated
queryStringDictionary.toString = function() {
if (queryStringDictionary.length == 0) {
    return "";
}
var toString = "?";
for (var key in queryStringDictionary) {
    toString += key + "=" +
    queryStringDictionary[key];
}
return toString;
};
// Return the key/value dictionary
return queryStringDictionary;
})();
```

Now you can parse for the individual params

```
// If "AccType" isn't there, accType is undefined
var accType = location.querystring["AccType"];
if(accType == "A")
{
    //Authorized
}
```