twoday

Ciceron Federation Services

TS-REST API

Version 1.0.8

1.0.8 – Add a .Net sample

1.0.7 - Add information about the errorObject returned for errors

1.0.6 - Add information about GetSession(). It cannot be called with the sessionid as a form value

1.0.5 - Describe personalNumber when signing

1.0.4 - relayState added and information about trusting callbackUrl

1.0.3 - Information about Security

1.0.2 - added Freja eID+ return values

1.0.1 - first release

Content

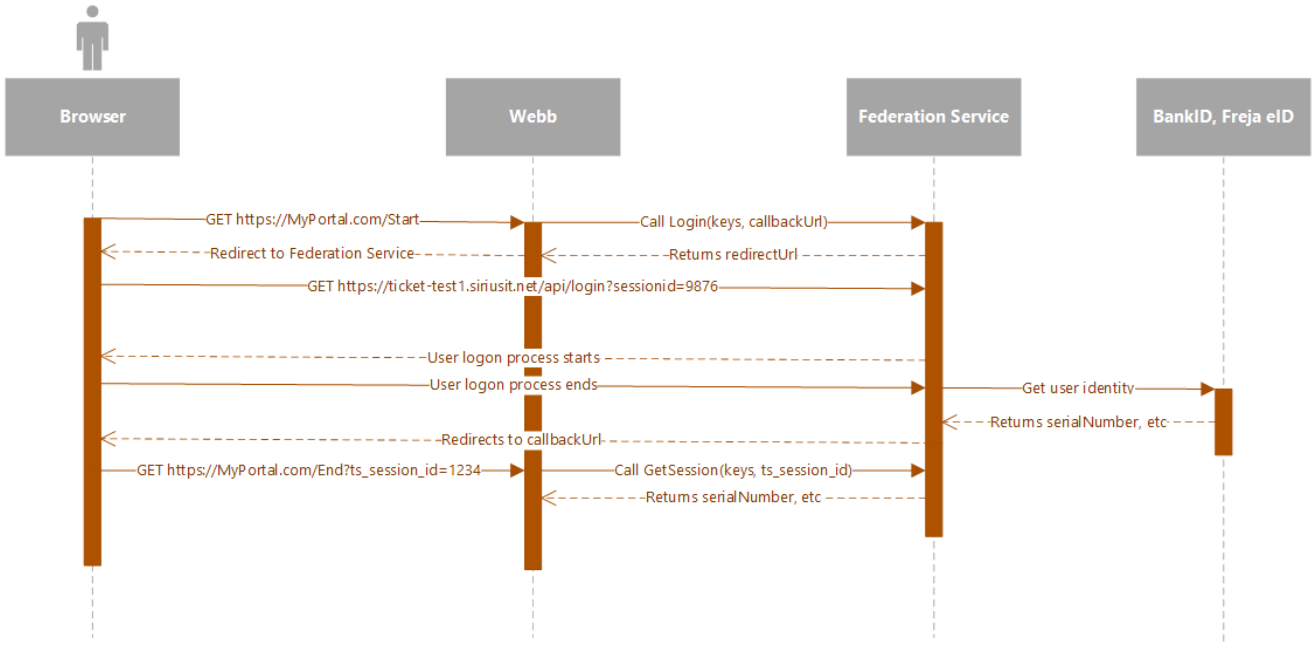
Introduction

Ciceron Federation Services is an identity portal for identification and signing. The main protocol for this service is SAML. The REST API is an alternative to SAML and can be used by all development platforms to secure web applications.

Before you begin development, you need a customer key and a service key to gain access to the testing and production environment.

When using the REST API for signing, it is important that you store the text to be signed and the signature result in order to be able to verify digital signatures when users deny the action.

Use case



Environments

Environment	Description
Test	Provided by *t*woday
Production	Provided by *t*woday

Glossary

Word	Description
API	Application Programming Interface
SAML	Security Assertion Markup Language is an open standard for exchanging authentication and authorization data between parties
cURL	Command-line tool (curl) for transferring data using various network protocols
JSON	JavaScript Object Notation is an open standard for data interchange

TS-REST API

Functions

Login

URL: **/json1.1/Login**

This function starts a logon request and returns a redirect url and session identity.

The following query strings must be added to the URL or posted as form data with a POST request:

Query string	Description	Demand
customerKey	Customer key generated and deployed by twoday support	mandatory
serviceKey	Service key generated and deployed by twoday support	mandatory
callbackUrl	URL for your web application, like https://myportal.com/login	mandatory
relayState	Customer information that will be returned in the callbackUrl	optional

Note:
The query strings and posted data must be URL encoded.

The callbackUrl used must be marked as trusted for the customer at twoday. Multiple URL:s can also be trusted (mainly in a test environment).

Example with cURL:

```
curl -k -X POST https://ticket-test1.siriusit.net/json1.1/Login?customerKey=1111&
serviceKey=2222" -H "accept: application/json" -H "Content-Type: application/x-www-
form-urlencoded" -d "callbackUrl=http%3A%2F%2Flocalhost"
```

returns:

```
{
  "redirectUrl":"https://ticket-test1.siriusit.net/api/login?
sessionid=8C118246A8A511DC3A7716548CC8116BBC21B078DC",
  "sessionId":"8C118246A8A511DC3A7716548CC8116BBC21B078DC"
}
```

When the browser redirects to the returned “redirectUrl”, the identification process begins. When it is completed, the browser is then redirected to the “callbackUrl” and query string “ts_session_id” is added. The “relayState” information (if specified) is also added as a query string to this URL.

GetSession

URL: **/json1.1/GetSession**

This function returns JSON information about the authenticated user or signature information. It is called from the “callbackUrl” specified in the Login or Sign request.

The following query strings must be added to the URL:

Query string	Description	Demand
customerKey	Customer key generated and deployed by twoday support	mandatory
serviceKey	Customer key generated and deployed by twoday support	mandatory
sessionId	The new ts_session_id from callback URL	mandatory
logout	If set to “true”, the session will be deleted after the call	optional

Note! The query strings must be URL encoded

Example with cURL:

```
curl -k "https://ticket-test1.siriusit.net/json1.1/GetSession?
customerKey=<yourkey>&
serviceKey=<yourkey>&sessionId=DA0AA89DFE4D674EC026A9CD2E054252FB3D8B9C58"
```

BankID return values:


```
{
  "sessionId": "DA0AA89DFE4D674EC026A9CD2E054252FB3D8B9C58",
  "userAttributes": {
    "C": "SE",
    "CN": "KALLE ANDERSSON",
    "GN": "KALLE",
    "O": "Testbank A AB",
    "SN": "ANDERSSON",
    "idp": "WPKI",
    "issuerCommonName": "BankID File",
    "name": "(190808 17.01) KALLE ANDERSSON - BankID",
    "serialNumber": "121212121212",
    "system": "Test",
    "type": "auth"
  },
  "username": "121212121212"
}
```

Freja eID+ return values:


```
{
  "sessionId": "DA0AA89DFE4D674EC026A9CD2E054252FB3D8B9C58",
  "userAttributes": {
    "C": "SE",
    "CN": "Johan Björklund",
    "G": "Johan",
    "SN": "Björklund",
    "dateOfBirth": "1966-01-08",
    "email": "johan.bjorklund@somewhere.com",
    "idp": "FREJA",
    "issuerCommonName": "Freja eID+",
    "serialNumber": "196601081234",
    "system": "Test",
    "type": "auth",
    "urn:oid:1.2.752.201.3.2": "dNVXnA8pQ_U9H-gdnFhj-xJYj08mSaTiXSvjsXfKrs-
9QKlAqW2HLciWHO5ZVbMW"
  },
  "username": "196601081234"
}
```


Note:

“BankID” will for some issued certificates return “GN” instead of “G” for givenName.

“Freja eID Extended” and “Freja eID+” will return userAttributes and serialNumber.

“Freja eID Basic” will only return “email”.

Sign

URL: **/json1.1/Sign**

This function starts a sign request and returns a redirect url and session identity.

The following query strings must be added to the URL or posted as form data with a POST request:

Query string	Description	Demand
customerKey	Customer key generated and deployed by twoday support	mandatory
serviceKey	Service key generated and deployed by twoday support	mandatory
callbackUrl	URL for your web application, like http://myportal.se/login	mandatory
userVisibleData	Text to be signed - must be Base64 encoded. BankID - max 40000 characters Freja - max 4096 characters Can also contain HTML to format the information in the web browser.	mandatory
userNonVisibleData	Text to be signed - must be Base64 encoded BankID - max 200 KB Freja - max 10 MB	optional
personalNumber	Prefill user identity for signing (a personal number, serialNumber or HSA id but not an email address or phone number)	optional
relayState	Customer information that will be returned in the callbackUrl	optional

Note:

The query strings and posted data must be URL encoded

The callbackUrl used must be marked as trusted for the customer at twoday. Multiple URL:s can also be trusted (mainly in a test environment).

Example with cURL:

```
curl -k -X POST https://ticket-test1.siriusit.net/json1.1/Sign?
customerKey=<yourkey>&ser
serviceKey=<yourkey>" -H "accept: application/json" -H "Content-Type:
application/x-www-form-urlencoded"
-d "callbackUrl=http%3A%2F%2Flocalhost&userVisibleData=VEVTVAo%3D"
```

returns:

```
{
  "redirectUrl":"https://ticket-test1.siriusit.net/api/sign?
sessionid=8C118246A8A511DC3A7716548CC8116BBC21B078DC",

  "sessionId":"8C118246A8A511DC3A7716548CC8116BBC21B078DC"
}
```

When the browser redirects to the returned “redirectUrl”, the identification process begins. When it is completed, the browser is then redirected to the “callbackUrl” and query string “ts_session_id” is added. The “relayState” information (if specified) is also added as a query string to this URL.
A call to GetSession() will return information about the signing, and the complete JSON result can then be stored together with the userVisibleData/userNonVisibleData.

URL: **/json1.1/Logout**

This function deletes the session.

The following query strings must be added to the URL or posted as form data with a POST request:

Query string	Description	Demand
customerKey	Customer key generated and deployed by twoday support	mandatory
serviceKey	Customer key generated and deployed by twoday support	mandatory
sessionId	The new ts_session_id from callback URL	mandatory

Note! The query strings and posted data must be URL encoded

Example with cURL:

```
curl -k -X POST "https://ticket-test1.siriusit.net/json1.1/Logout?
customerKey=<yourkey>&
serviceKey=<yourkey>&sessionId=DA0AA89DFE4D674EC026A9CD2E054252FB3D8B9C58
```

returns:

```
{
  "sessionDeleted":1
}
```

Returned information

Login session

The following JSON attributes can be returned for login requests:

Data	Description
serialNumber	Users personal identity number
CN	Complete name
G or GN	Given name
SN	Surname
issuerCommonName	Issuer
type	Value auth for logon requests

Sign session

The following JSON attributes can be returned for sign requests:

Data	Description
serialNumber	Users personal identity number
CN	Complete name
G or GN	Given name
SN	Surname
certificate	Base64 encoded certificate
signature	Base64 encoded signature
transactionid	Unique id for this signing request
timestamp	Time for this signing request
signRef	Signature reference. A hashkey of the TimeStamp and the SignDigest
signMessage	Visible text to be signed. Contains a TimeStamp and SignRef
signDigest	Hash value of text to be signed. A Base64 encoded string of the userVisibleData digest
type	Value sign for sign requests

Note:

An application that wants to digital sign information should store this JSON-data together with userVisiblData and userNonVisibleData to be able to verify the signed information.

Sample

Simple .Net sample. Make sure to check returned responses. JSON will only be returned if the server has responded. A proxy can for instance return “502 Bad gateway” or some other errors. Check that the returned “Content-Type” is “application/json” to be able to parse it as JSON.

```
using System.Net;
using System.Text.Json;
using Microsoft.AspNetCore.Http.Extensions;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;

namespace Demo.Pages
{
#pragma warning disable CS8618
    public class CFSError
    {
        public string code { get; set; }
        public string message { get; set; }
    }

    public class CFSUserAttributes
    {
        public string serialNumber { get; set; }
        public string GN { get; set; }
        public string SN { get; set; }
        public string O { get; set; }
    }

    public class CFSResponse
    {
        public CFSError errorObject { get; set; }
        public string redirectUrl { get; set; }
        public CFSUserAttributes userAttributes { get; set; }
        public string username { get; set; }
    }
#pragma warning restore CS8618

    public class LoginModel : PageModel
    {
        public string  Message = "";
        private readonly string loginEndpoint = https://idp.ciceron.cloud/json1.1/Login?customerKey={0}&serviceKey={1}&callbackUrl={2};
        private readonly string sessionEndpoint = https://idp.ciceron.cloud/json1.1/GetSession?customerKey={0}&serviceKey={1}&sessionId={2};
        private readonly IConfiguration config;

        // Setup JSON configuration by appsettings.json for customer and services keys
        public LoginModel(IConfiguration configuration)
        {
            config = configuration;
        }
```

```
        // Build in handler for HTTP GET /Login
        public async Task<IActionResult> OnGet(string ts_session_id)
        {
            if (ts_session_id == null)
                return await BeginAuthenticate();
            else
                return await EndAuthenticate(ts_session_id);
        }

        // Start the logon processs with twodays magic services
        private async Task<IActionResult> BeginAuthenticate()
        {
            var api = new HttpClient();
            string result = await api.GetStringAsync(String.Format(loginEndpoint,
config["customerKey"],  config["serviceKey"], Request.GetDisplayUrl()));

            var response = JsonSerializer.Deserialize<CFSResponse>(result);

            if(response == null)
                Message = "Technical error";
            else if (response.errorObject != null)
                Message = response.errorObject.message;
            else
                return Redirect(response.redirectUrl);

            return Page();
        }

        // End the logon processs and identify the user
        private async Task<IActionResult> EndAuthenticate(string id)
        {
            var api = new HttpClient();
            string result = await api.GetStringAsync(String.Format(sessionEndpoint,
config["customerKey"], config["serviceKey"], id));

            var response = JsonSerializer.Deserialize<CFSResponse>(result);

            if (response == null)
                Message = "Technical error";
            else if (response.errorObject != null)
                Message = response.errorObject.message;
            else
                Message = response.userAttributes.serialNumber + " " +
response.userAttributes.GN + " " + response.userAttributes.SN + " " +
response.userAttributes.O;

            return Page();
        }
    }
}
```

Common errors

If an “errorObject” is returned it will contain a “code” and a “message” containing the error.

```
{
  "errorObject": {
     "code": "error code example",
     "message": "Information about the error"
  }
}
```

The following are examples of common errors:

code = “NOTLOGGEDIN”
The user is not identified or the user has canceled the login or sign.

Failed to use GetSession

You are probably using the wrong sessionId, you must use the session identity returned in the callback URL (`ts_session_id`) instead of the one in the `redirectUrl`.

Failed to use Login

You are probably using the wrong customer key or service key.

The specified callbackUrl must also be trusted at twoday.

Contact twoday support for more information.

Security

Keys

The customerKey and serviceKey is ordered from twoday ServiceDesk, ciceron@twoday.com. twoday will only deliver the customerKey and serviceKey to the correct customer contact using a secure channel.

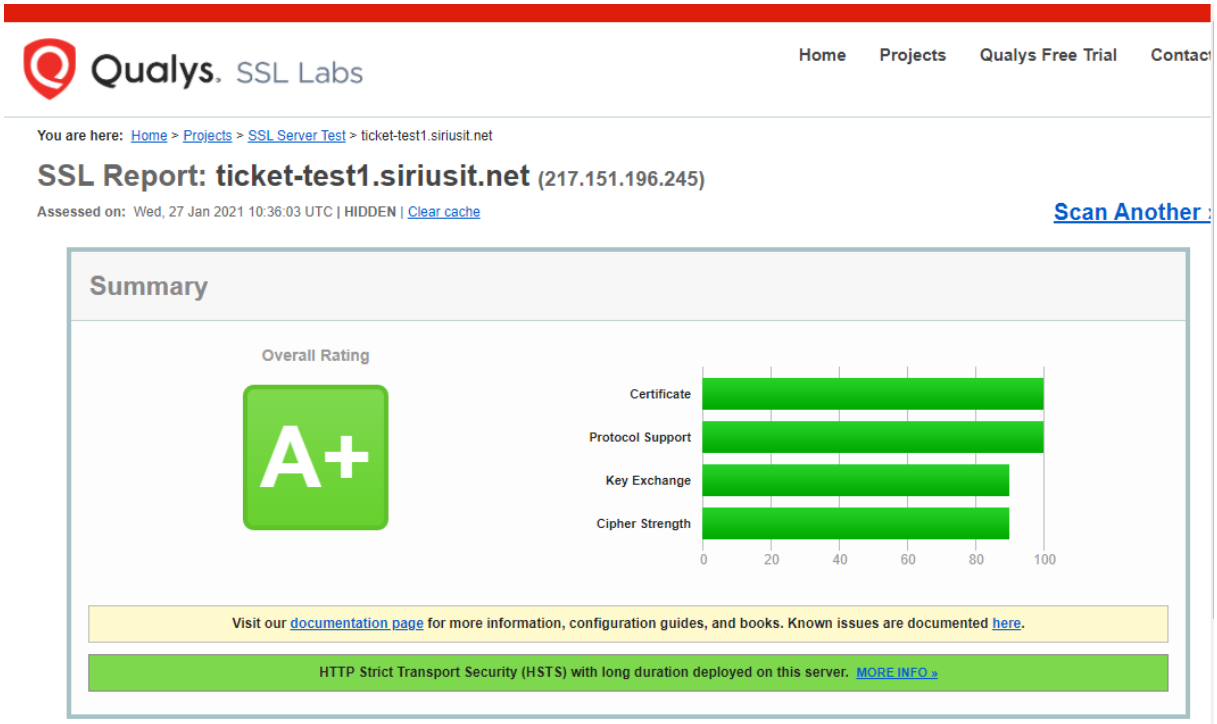
The keys are unique per customer and service and must not be shared by others and configured and installed in a secure environment.

Endpoint

The REST-API endpoint is installed and configured according to

- Certifikat SHA256 Activated
- Port 443 (SSL) Activated
- TLS 1.2 Activated
- Strict Transport Security (HSTS) Activated

The endpoint is also supervised with Qualys and must be rated as A+ according to



Trusted Site

The callback URI for the customer must also be configured in the endpoint for twoday REST-API as an allowed trusted site. This means that other sites will not be able to use the REST-API without a correct configuration.

Certificates

It is also possible to activate a client certificate for a customer accessing the REST-API. This means that the customer creates a certificate used for accessing the REST-API.

In this case a new endpoint URI for the REST-API must be used.

IP-blocking

It is also possible to IP-block for accessing the REST-API. This means that only the configured IP-addresses can be used for accessing the REST-API.

In this case a new endpoint URI for the REST-API must be used.