
Article

[Dmitry Maslennikov](#) · May 11, 2020 5m read

[Open Exchange](#)

Work with SAML in IRIS

When a company is quite large and many different applications used by employees. But while those applications are mostly completely different, how to make it possible to not force users to enter credentials as many times as many applications they would like to use. The best way is to use SSO, so, it will be possible to have a portal, where users could launch any application used in a company. There are many different ways how to give access to your application by using the SSO mechanism, and some of them are:

- OAuth2
- Kerberos
- SAML

InterSystems already supports [OAuth2](#) and can be quite easily deal with Kerberos. But I would like to discuss about using [SAML](#) (Security Assertion Markup Language).

SAML is an open standard for exchanging authentication and authorization data between parties, in particular, between an identity provider and a service provider.

In authentication with SAML participating three parties:

- Service Provider (SP) - web-based application. In our case, it will be our IRIS based application
- Identity Provider (IdP) - any external software with a role as Identity Provider. It can be
 - G-Suite
 - Microsoft Azure, myapps.microsoft.com
 - Shibboleth
 - OneLogin
 - and so on
- User Agent - browser where your user use to work with your application

SAML terminology

Before we start to speak how to use SAML, in your application, add just a few words about terms, used in SAML in the context of using in the authentication process.

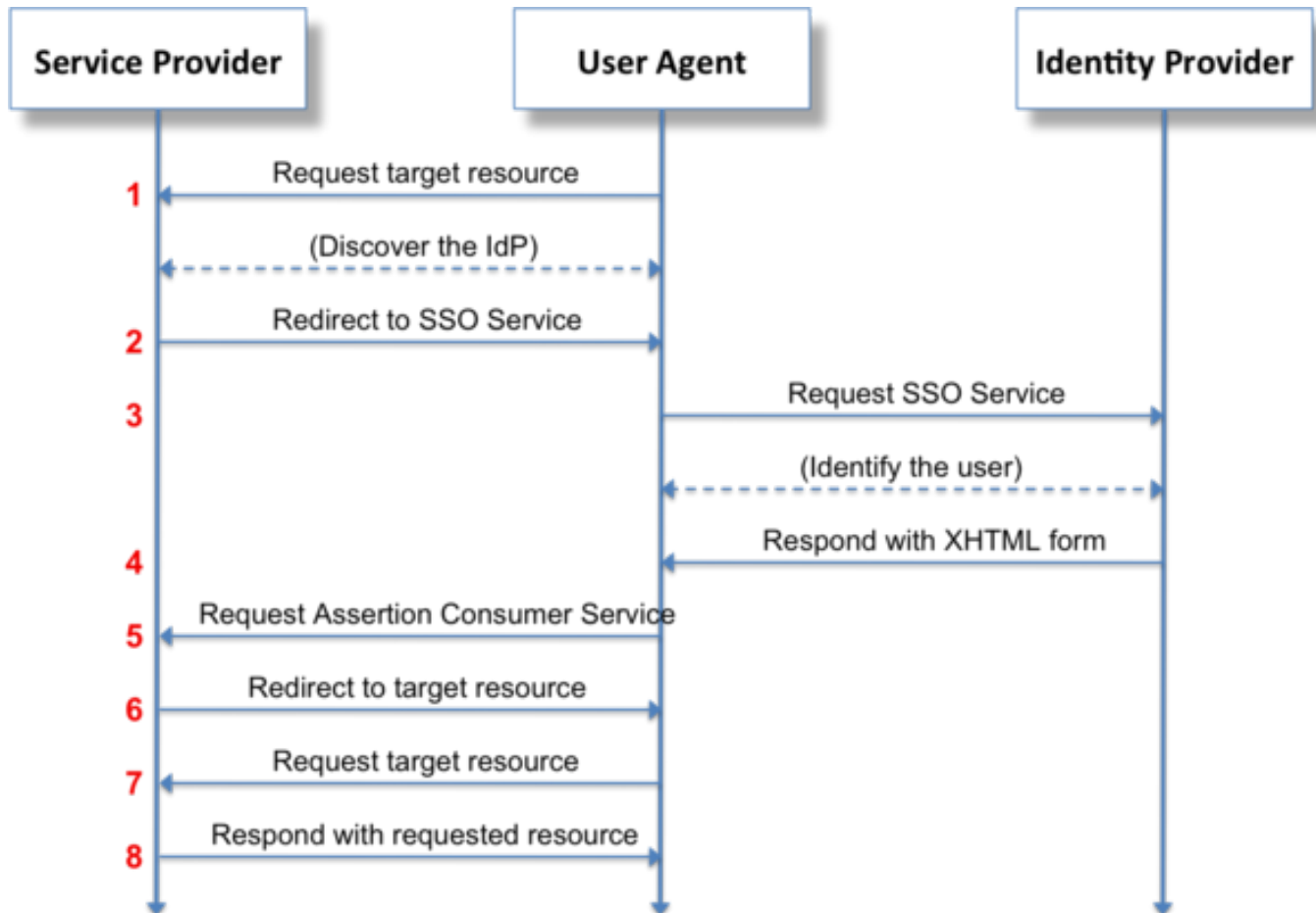
- Service Provider (SP) - is a system entity that receives and accepts authentication assertions in conjunction with a single sign-on (SSO) profile
- Identity Provider (IdP) - is a system entity that issues authentication assertions in conjunction with a single sign-on (SSO) profile
- SAMLRequest - a message in form of XML, prepared by Service Provider and sent to Identity Provider
- SAMLResponse - a message in form of XML, prepared and signed by Identity Provider and sent to Service Provider
- EntityID - some unique value for SP or IdP, which identifies this particular instance, in most cases in the form of URI
- ACS URL, Assertion Consumer Service URL - URL on SP side, which can process queries with parameter SAMLResponse, should validate SAMLResponse and create a session for the user
- Certificate - issued and used by IdP, to sign SAMLResponse. Service Provider should store issued by IdP certificate, to validate SAMLResponse
- IdP Login/SSO URL - URL on Idp side, accepts queries containing SAMLRequest, validates and redirects

to the authentication page if needed, and redirects back to SP

- IdP Metadata - XML-based metadata contains some settings from IdP, such as SSO URL, and Certificate

Interaction

The interaction scheme itself on the picture



And there are two cases of interaction

1. The user opens your application directly, for the first time in the current session. The application prepares a message for Identity Provider, and redirects to Login Url, when processed authentication, and redirects back with a specially formed response.
2. First of all, the user goes to Identity Provider's portal, already authorized, and opens desired our application, and IdP forms URL with the response and redirects the user to the application.

Practice

From theory go to practice. I've published the new repository, <https://github.com/daimor/iris-saml-example> which contains an example of how it can be used in InterSystems IRIS. It can be started with docker. In my examples below I will show how to use it with GSuite, with any other Identity Provider, settings should be very similar, just keep in mind some caveats, described at the end of this article.

Start a simple Service Provider, in this role will be my example. So, just clone this repo, and start it with docker-compose

```
$ git clone https://github.com/daimor/iris-saml-example.git
$ cd iris-saml-example
```

```
$ docker-compose up -d --build
```

After a while after build will be completed, and IRIS will start, our test application will be available by URL <http://localhost:19092/csp/irisapp/MyApp.cls>. Google and some other IdP may require that your application should be used over HTTPS. To have access over https to my local application I'm using [ngrok](#) project. Let's start secure tunnel to our application.

```
$ ngrok http 19092
```

It should output something like this.

```
Session Status      online
Account             Dmitry Maslennikov (Plan: Free)
Version             2.3.35
Region              United States (us)
Web Interface        http://127.0.0.1:4040
Forwarding           http://b571ff3e.ngrok.io -> http://localhost:19092
Forwarding           https://b571ff3e.ngrok.io -> http://localhost:19092

Connections         ttl    opn    rt1    rt5    p50    p90
                   11     0      0.00   0.00   0.01   234.16

HTTP Requests
```

So, our application now available by link <http://b571ff3e.ngrok.io/csp/irisapp/MyApp.cls>, and should show such page

SAML Authentication Example

Use these settings to configure Identity Provider

Login URL, ACS Url:

EntityId: *https://intersystems.com/saml/F53FF87C-7B43-11EA-B44C-0242C0A80101*

Identity provider Metadata file: No file chosen

The next step will be to configure IdP, in my case G-Suite, go to [Admin Console](#), Apps, SAML Apps, SETUP MY OWN CUSTOM APP

Step 2 of 5

Google IdP Information

Choose from either option to setup Google as your identity provider. Please add details in the SSO config for the service provider. [Learn more](#)

Option 1

SSO URL: `https://accounts.google.com/o/saml2/idp?idpid=C03hrevd1`

Entity ID: `https://accounts.google.com/o/saml2?idpid=C03hrevd1`

Certificate: **Google_2025-4-11-134116_SAML2.0**
Expires Apr 11, 2025

[Download](#)

OR

Option 2

IDP metadata: [Download](#)

[PREVIOUS](#) [CANCEL](#) [NEXT](#)

Where google issued a new certificate for our new application. We need IDP Metadata file.

Next, define the name of the application

Step 3 of 5

Basic information for your Custom App

Please provide the basic information needed to configure your Custom App. This information will be viewed by end-users of the application.

Application Name *: app-id: irisapp

Description:

Upload logo: [Choose File](#)

This logo will be displayed for all users who have access to this application. Please upload a .png or .gif image of size 256 x 256 pixels.

[PREVIOUS](#) [CANCEL](#) [NEXT](#)

Then we should fill ACS URL and Entity ID with values from our application. And select Name ID Format, EMAIL, or any other value, which can be used as a login in your system.

Step 4 of 5

Service Provider Details


Please provide service provider details to configure SSO for your Custom App. The ACS url and Entity ID are mandatory.

ACS URL *	<input type="text" value="https://b571ff3e.ngrok.io/csp/irisapp/MyApp.cls"/>		
Entity ID *	<input type="text" value="https://intersystems.com/saml/F53FF87C-7B4311E"/>		
Start URL	<input type="text"/>		
Signed Response	<input type="checkbox"/>		
Name ID	<input type="text" value="Basic Information"/>	<input type="text" value="Primary Email"/>	
Name ID Format	<input type="text" value="EMAIL"/>		

PREVIOUS CANCEL NEXT

On the next step, just finish the configuration.

And finally, we should enable it, for the users in our organization

 IRISAPP

All users in this account

Groups

Organisational units


Showing settings for users in all organisational units

Service status:

Service status:

☒ ON for everyone

☐ OFF for everyone

 Changes may take up to 24 hours to propagate

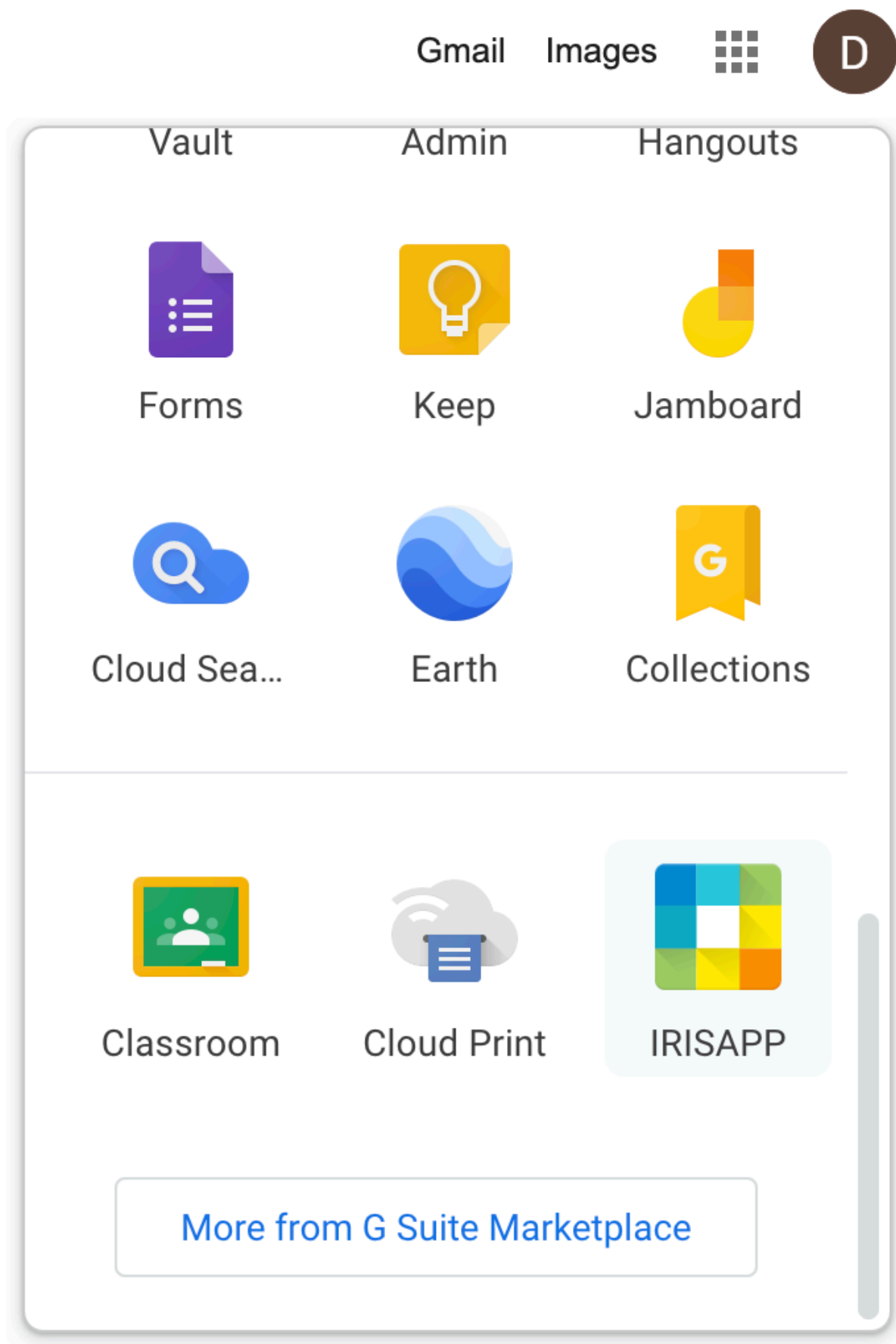
Return back to our application, in the file field choose previously downloaded metadata file, and press submit.

ACS Url: <https://b571ff3e.ngrok.io/csp/irisapp/MyApp.cls>
SSO Url: <https://accounts.google.com/o/saml2/idp?idpid=C03hrevd1>
Entity Id: <https://accounts.google.com/o/saml2?idpid=C03hrevd1>
nameIDFormat: urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress
Certificate:

Page 6 of 9

should keep in mind that IRIS does not validate SAMLResponse if only Response signed it returns ERROR #6390: Signature validation failed: No Assertion.

So, our application is now available from the list of Google applications



user's info, so then you can use own mechanism or [delegated authorization](#).

Caveats

- Namespace should be Ensemble enabled. Validation of SAML lies in class Ens.Util.XML.SecuritySignature
- InterSystems IRIS (tested version 2020.2) and below does not validate if only Response signed, and requires Assertion to be signed.
- SAMLResponse sent by URL's query, while encoded with Base64 can be too long, and in some cases truncated. Some IdP supports Response and Assertion signed at the same time, so, recommend switching off signing Response.
- SAMLResponse and SAMLRequest by standard support compression before encoding with Base64. IRIS does not support such compression directly.

[#Access control](#) [#Authentication](#) [#InterSystems IRIS](#)

[Check the related application on InterSystems Open Exchange](#)

Source URL: <https://community.intersystems.com/post/work-saml-iris>