Article <u>Dmitry Maslennikov</u> · 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 <u>OAuth2</u> and can be quite easily deal with Kerberos. But I would like to discuss about using <u>SAML</u> (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

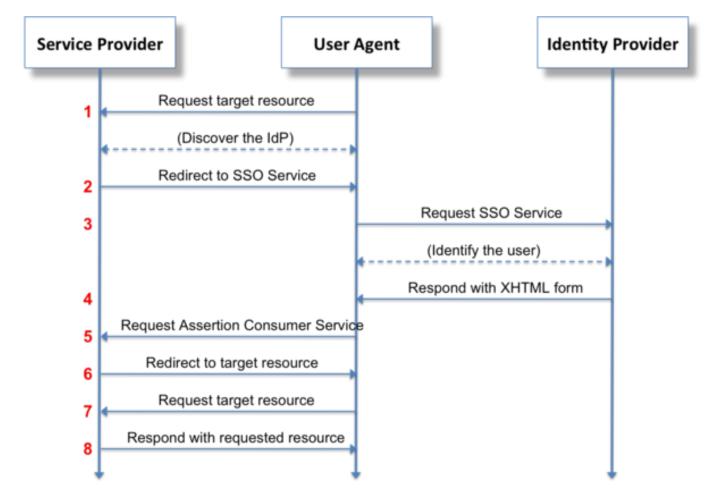
Before we start to speak how to use SAML, in your application, add just a few words about terms, used in SAML in the context of using in the authentication process.

- 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, <u>https://github.com/daimor/iris-saml-example</u> which contains an example of how it can be used in InterSystems IRIS. In 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 dockercompose

```
$ 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 <u>http://localhost:19092/csp/irisapp/MyApp.cls</u>. 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 <u>ngrok</u> project. Let's start secure tunnel to our application.

\$ ngrok http 19092

It should output something like this.

Session Status Account Version Region Web Interface Forwarding Forwarding	2.3.3 Unite http: http:	y Maslen 5 d States //127.0. //b571ff	0.1:4040 3e.ngrok.	.io -> ht	tp://loc	alhost:19092 calhost:19092				
Connections	ttl 11	opn Ø	rt1 0.00	rt5 0.00	p50 0.01	p90 234.16				
HTTP Requests										

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

SAML Authentication Example

Use these settings to configure Identity Provider

```
Login URL, ACS Url: [https://b571ff3e.ngrok.io/csp/irisapp/MyApp.cls
EntityId: https://intersystems.com/saml/F53FF87C-7B43-11EA-B44C-0242C0A80101
```

Identity provider Metadata file: Choose file No file chosen

Submit Clear

The next step will be to configure IdP, in my case G-Suite, go to <u>Admin Console</u>, Apps, SAML Apps, SETUP MY OWN CUSTOM APP

Status	Certificate
Step 2 of 5 Google IdP I	Information 5-4-11-134116_SAML
	er option to setup Google as your identity provider. Please add details in the SSO ice provider. Learn more
Option 1	5-4-11-134116_SAML 11, 2025
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

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

	on for your Custom App		× :5-4-11-134116_SAML2.0 11, 2025
viewed by end-users of		ur Custom App. This information will be	5-4-11-134116_SAML2.0
Application Name *	IRISAPP	app-id: irisapp	11, 2025
Description			
Upload logo	U CHOOSE FILE		
	This logo will be displayed for all us	sers who have access to this application.	
	Please upload a .png or .gif image o	of size 256 x 256 pixels.	

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.

Next, define the name of the application

Status 🔺				Certifics	ato
Step 4 of 5 Service Provide	er Details			×	.5-4-11-134116_SAM 11, 2025
Please provide service ID are mandatory.	provider details to configure S	SO for you	r Custom App. The A	CS url and Entity	
ACS URL *	https://b571ff3e.ngrok.io	/csp/irisap	pp/MyApp.cls		:5-4-11-134116_SAM 11, 2025
Entity ID *	https://intersystems.com	/saml/F53	FF87C-7B43		
Start URL					
Signed Response					
Name ID	Basic Information		Primary Email	~	
Name ID Format	EMAIL	~			

On the next step, just finish the configuration.

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

		Showing settings for users in all organisational units				
All users in this account		Service status:				
Groups	~	Service status: ON for everyone				
Organisational units	^	OFF for everyone				
		(i) Changes may take up to 24 hours to propagati				

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

Identity provider

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:** ----BEGIN CERTIFICATE----

MIIDdDCCAlygAwIBAgIGAXFt+rjRMA0GCSqGSIb3DQEBCwUAMHsxFDASBgNVBAoTC0dvb2dsZSBJ
bmMuMRYwFAYDVQQHEw1Nb3VudGFpbiBWaWV3MQ8wDQYDVQQDEwZHb29nbGUxGDAWBgNVBAsTD0dv
$b2ds {\it ZSBGb31gV29yazELMAkGA1UEBhMCVVMxEzARBgNVBAgTCkNhbGlmb3JuaWEwHhcNMjAwNDEy}$
MTA0MTE2WhcNMjUwNDExMTA0MTE2WjB7MRQwEgYDVQQKEwtHb29nbGUgSW5jLjEWMBQGA1UEBxMN
TW91bnRhaW4gVmlldzEPMA0GA1UEAxMGR29vZ2x1MRgwFgYDVQQLEw9Hb29nbGUgRm9yIFdvcmsx
CzAJBgNVBAYTA1VTMRMwEQYDVQQIEwpDYWxpZm9ybmlhMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8A
MIIBCgKCAQEAv51aR3RIKj5P7Mn296XlJkmiQAqAhIGSpvMH/6pXTMM8sJwXSpanxUaAUsjZ/msE
00aa1qNcwAb0qam1nddR77bhVzcTeQ0GL7YY88R1KpRzPQ2ukLBhCfD++6xPIa6cjMVIJM8NZCug
SW0TDkjdd8vFCJMJEppnmeAoife0H0TkhTrSXWNzHwT5i2mvLHBViMwbnt825rLidX50EoAl3x2D
mWODqaVOcAAr34j36cMi/7nhqz+Pq138UBBuecmxxtv7/x1cJf94X3KyZkjD71LOSjaTds2nFqQQ
$aDkty 17 \\ Yu \\ Y5 \\ 1mD \\ 1mSG0 \\ jV \\ p14 \\ Ru \\ jOr \\ AN \\ we \\ 6X \\ KA \\ UV \\ Tv \\ 1mH \\ 2i \\ cx \\ PQ \\ IDA \\ QABMA \\ 0G \\ CS \\ qGS \\ IDS \\ DQ \\ EB \\ CW \\ AN \\ SS \\ SS \\ SS \\ SS \\ SS \\ SS \\ SS$
A4IBAQBrdeClT5WKr9Y7AasOup709a9D0Jbgy23VM9yZ31r2cUpRUe8H/afS0HijfIbVn5CNOEh3
70mF+1PKgx2Fa74NghJTcmoR073bz1ae/91PXfdMhIRsk0KqSbi1AnYYu0yFMtCAVi0vMRJsX5Mt
ery55tDhE6DXwb0MABk07h0zM/HMX/tbnYWJk6jP9H5+Xg6rOcKUKgJQPLuXDK0pDKyyMvBIY6N7
Xk11GfT7qj3P8iMRE17AF9U9pUF18fcAqbnlWPK9SI19n60CQNj1YntZlfBs2iIscdDFCvXTjY2m
bBLT9vMzWqLGHp9ZqpaE8HKHNNVDqeZmkPVQ+U4oh2zW
END CERTIFICATE

It should parse that file, and store it in our format on the IRIS side.

Below it contains a simple SAMLRequest, which will be used when you press the Login button.

SAMLRequest
<pre><?xml version="1.0" encoding="UTF-8"?> <samlp:authrequest assertionconsumerserviceurl="https://b571ff3e.ngrok.io/csp/irisapp/MyApp.cls" destination="https://saccounts.google.com/o/saml2/idp:idpid=C03hrevd1" id="_4be61246920611eabcd20242ac170002" ispassive="false" issueinstant="2020-05- 09715(31:52" version="2.0" xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion" xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"></samlp:authrequest></pre>
Login

By Login button, it makes a POST a form with SSO URL provided by Identity provider as action URL, and SAMLRequest field which contains an XML from above encoded as Base64.

Identity Provider, should validate our SAMLRequest, offer form login form if the user not authorized yet. And redirect back. So in our application, we should see something like this. SAMLResponse



Validation: Success NameID: dmitry.maslennikov@caretdev.com

Here shown SAMLResponse provided by IdP, and it contains a signature, which has to be validated with the certificate which we got from the metadata file. When SAMLRequest validated, it shows the status and NameID (email) of the user.

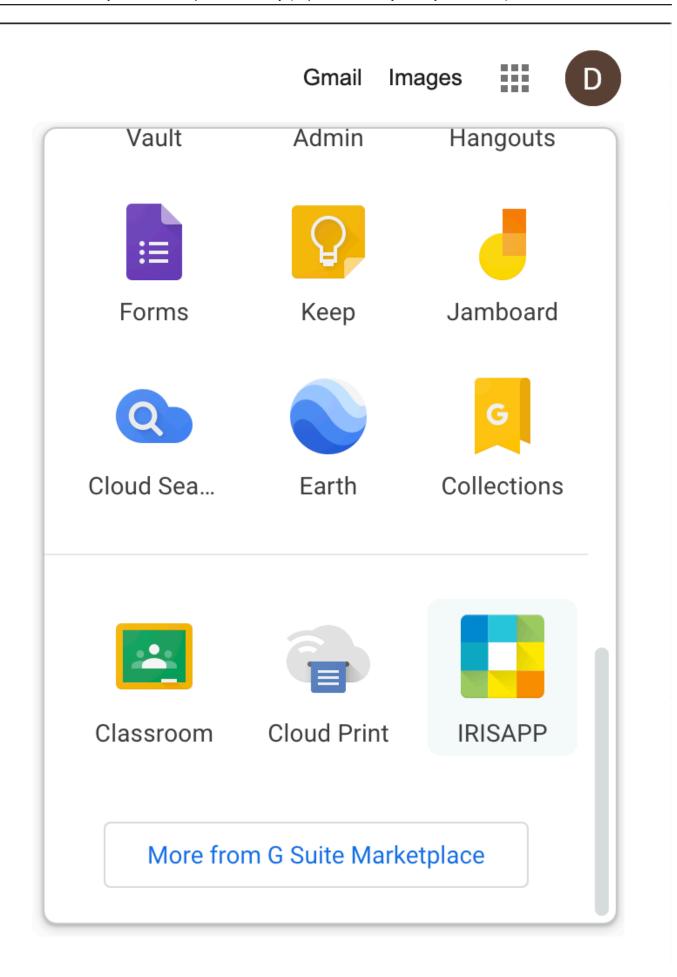
Validation may fail with the error like this:

Validation: ERROR #6390: Signature validation failed: Failed NotBefore/NotOnOrAfter (/2020-05-09T15:17:48.525Z) date time verification with clock skew of 0 seconds.

SAMLResponse can contain one or two signatures. The Assertion or the whole Response can be signed. But you

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



This article does not contain information on how to authorize users in your application, it just shows how to retrieve

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