Inbound SSO to Korero

Single Sign-On (SSO) allows a user to access multiple web applications using one set of credentials, meaning they don’t need to remember multiple usernames and passwords. It also helps from a security standpoint and saves time/effort in scenarios where companies want to ‘link’ separate systems whilst maintaining just a single set of users (as opposed to duplicating user detail management across multiple systems).

There are different types of SSO. This documentation covers the scenario where another system is the Identity Provider (i.e. they store the credentials and handle authentication, however can gain access to Korero by logging into that system first, then being redirect to Korero).

Inbound SSO

Federated Authentication allows another authentication provider, such as Active Directory (on-prem or cloud-based), to authenticate users accessing Korero. With Federated Authentication:

* A separate system is the primary Identity Provider
* Korero acts as the Service Provider

Types of Requests

We currently support SSO into Korero using:

* SAML / SAML2
* OAuth2
* JWT

So far the vast majority of Federated Authentication integrations we’ve had running have been SAML-based using Active Directory (usually hosted on Azure).

### Configuration Details

In order to connect Korero as a Service Provider to an Identity Provider, we need to set configuration values inside Korero so we know where to check authentication details etc.

Below is a table outlining the configuration values required:

|  |  |  |  |
| --- | --- | --- | --- |
| **Key** | **Description** | **Data Type** | **Example** |
| ADFSClaimIdentifier | The identifier/type for the claim we receive from the identity provider | Text | Email |
| ADFSIdentifier | An identifier we declare and share with the identity provider – this gets encoded in query string for the login url as “client\_id” for OAuth. For SAML it is set as the “AssertionConsumerServiceURL” attribute | Text | https://acme.korero.io/api/sso/callback |
| ADFSLoginUrl | The identity provider’s endpoint for the login request | Text | https://fs.acme.com/adfs/ls/ |
| ADFSLogoutUrl | The identity provider’s endpoint for the logout request | Text | https://fs.acme.com/adfs/ls/?wa=wsignout1.0 |
| ApplicationIdentifier | The client id we send when retrieving a token | Text | Korero |
| SSOClaimMatchField | What we use to match the auth token with  Korero Username = 1 (e.g. jsmith25)  EmailAddress = 2  Employee Number/Ref = 3  SSO Claim = 4 | Integer | 2 |
| SSOIssuer | A valid issuer that will be used during token validation (ADFS trust) | Text | http://fs.acme.com/adfs/services/trust |
| **Key** | **Description** | **Data Type** | **Example** |
| SSOPureSAMLClient | Indicates if a SAML client follows the standard fully. If not they probably use ADFS. | Boolean | True |
| SSOType | Type of SSO used  SAML = 2  OAuth = 3  JWT = 4 | Integer | 2 |

### Summary

We need to discuss and agree with our client/technology partner:

1. Which type of SSO we’ll be implementing with them
2. Which employee property (e.g. employee number, email address) we’ll be using to match users

We then need to exchange some details, for SAML implementations this is:

1. Korero need to be given the login/logout urls, a claim identifier, a SAML version number, an Issuer value
2. Korero need to provide the third party ACS url to the identity provider partner
3. The identity provider partner needs to provide the SAML signing certificate, ideally as a Base64 string.

### User Matching

In order to authenticate a user, we need to match the authenticated user in the Identity Provider with a Korero user. A unique identifier of some form is needed which is agreed at the start of the SSO project (normally email address or employee reference number) then a look-up is performed on the set of Korero users for a single user with that matching identifier. If a match is found, authentication can be completed. If a match is not found, the user will be redirected to a manual login page for Korero.

### Testing

The configuration values provided to us can be set up in one of our test environments for testing the authentication process, and then copied across to production when it’s ready to go live.