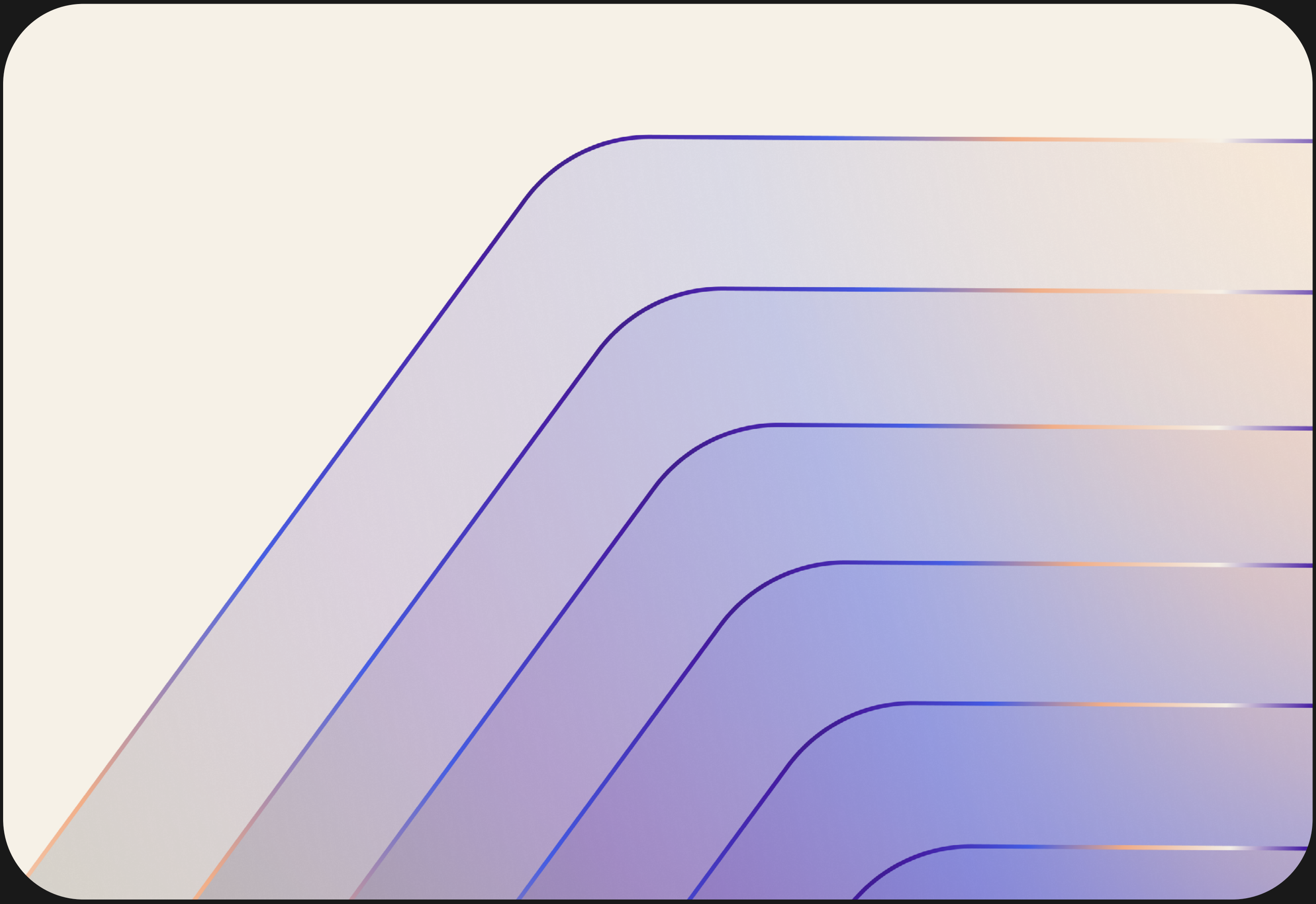
Whitepaper

Auth0’s guide to Single Sign-On (SSO)

A hands-on exploration of one of the most valuable capabilities in Identity and Access Management (IAM)



okta

Auth0 by Okta

Contents

Introduction

Thank you for downloading Auth0's Updated Guide to Single Sign-On (SSO)

Whether you're learning about SSO for the first time, wishing to catch up on the latest trends and best practices, or are in the process of implementing Single Sign-On within your app or organization, this guide is for you.

We'll touch upon many authentication and Identity topics with the understanding that you, the reader, have some general technical knowledge and experience. Of course, we'll also include many references and resources should you wish to dive deeper into various topics.

The goals of this guide are to help you to:

- Understand what Single Sign-On is and how it works,
- Know the benefits of using Single Sign-On within your organization,
- Learn why potential customers value (or even require) SSO support,
- Become familiar with the various Identity protocols used in conjunction with SSO,
- Implement Single Sign-On in a guided tutorial.

We have a lot to cover — so let's get started!

Introduction to Single Sign-On (SSO)

Regardless of your role or position, if your job involves using software then you've likely encountered Identity and authentication. Maybe you were a developer in charge of securing an application or were a frustrated end user having to remember yet another set of credentials and thinking, “Isn’t there a better way?”

As our professional and personal lives move online (or at least depend upon online resources), proper Identity and Access Management (IAM) is becoming increasingly important — and Single Sign-On (SSO) has a significant role to play.

SSO, in a nutshell

Single Sign-On describes an Identity solution that allows users to use the same set of credentials for multiple applications, thereby avoiding the need for a user to interactively authenticate within each app. From the user’s perspective, they enjoy the convenience of logging in to one system and being automatically granted access to others.

You have likely come across SSO before, even if you didn't recognize it for what it was. Take Google, for example:

- Upon logging in to one Google service, such as Gmail, you are automatically authenticated to YouTube, AdSense, Google Analytics, and other Google apps.
- Likewise, if you log out of your Gmail or other Google apps, you are automatically logged out of all the apps.

SSO is an enterprise imperative

Single Sign-On is typically found in enterprise environments where employees access numerous apps and services on a daily basis.

The software-as-a-service (SaaS) model allowed for many companies to build specialized products that focused on solving a particular problem very well, transforming corporate technology stacks in the process. For example, rather than running an on-premises monolithic HR application to handle multiple areas such as payroll, employee information, and expense reports, a modern organization is likely to have a SaaS application that handles payroll, a SaaS app that handles employee information, and another SaaS app that allows employees to enter their expenses.

For context, Okta's Businesses at Work 2024 report indicates that the average Okta enterprise customer has deployed more than 200 apps, and even across companies of all sizes the average is more than 90!

Using Single Sign-On helps to ensure that security policies are consistently applied across all those applications, since the policies can be enforced at the enterprise's Identity Provider (IdP).

Plus, with fewer credentials for users to remember, there are fewer credentials that can be stolen, shared/sold, and abused by threat actors. Even partially reducing the risks associated with stolen credentials is significant, as they continue to be relied upon by attackers. For context, Verizon's 2024 Data Breach Investigations Report (DBIR) indicates that over 80% of breaches involve compromised Identity.

As enterprises continue to adopt best-of-breed tools, it's a reasonable expectation that IT leaders will increase their focus on access management strategies, like SSO, that can simultaneously enhance both security and the user experience.

With the stage set, let's briefly examine SSO's significant role in modern Identity.

SSO and modern Identity

In this section, we'll quickly explain where SSO fits into modern Identity and Access Management, before touching on SSO's relationship with Identity Federation.

(If you're already familiar with these subjects, feel free to skip ahead to the next section.)

Authentication, authorization, and Identity management

Authentication plays a pivotal role in verifying that users are who they say they are, making it an important function of modern IAM implementations.

However, authentication is just one piece of a bigger picture — authorization and Identity management are also essential.

Authentication

Authentication deals with verifying user Identity. Essentially, the authentication step attempts to establish with confidence that the users logging into accounts are who they say they are, using one or more factors to do so.

An idealized authentication system provides infinite friction for malicious actors and something near zero for genuine uses (because a little bit of friction in the right place at the right time can help to build trust).

While such a solution is a worthy objective, the real world frequently involves tradeoffs between friction and convenience. For example, think of all the password rules you've encountered over the years: length, complexity, the need to change them — these are all intended to create friction for attackers, but they cause problems for users.

Single Sign-On is attractive because it can provide a simple user experience while providing strong security measures that can be implemented across an entire organization. Since in an SSO implementation there is a single source of Identity, IT administrators can enforce strong authentication without overly inconveniencing users.

Authorization

Authorization deals with ensuring that users have the appropriate levels of access to privileges, resources, features, (etc.) within an application.

Once a user is successfully authenticated, they are given certain permissions for what they can do. For example, in a payroll application an accountant is able to process payroll and see details for all of the employees, while an individual employee is restricted such that they can only see and update their own details.

Ensuring the right level of access is very important and is difficult to do in a non-SSO implementation. Keeping track of who has access to what is likely to keep IT administrators very busy. Different applications also present different levels of granularity when it comes to user permissions — making managing authorization even more difficult.

Single Sign-On aims to solve the authorization component of modern Identity by providing a centralized source of truth for each individual user. Once a user is authenticated, the Identity Provider can send the user's Identity data and permissions to each application.

By providing IT administrators with a single location where they can manage user roles and permissions for the entire organization, SSO helps to simplify management, reduce errors, and enable detailed audits.

Identity management
Identity management is concerned with creating, updating, and deleting users as they move throughout an organization.

Let's look at Identity management through the lens of an employee:

- On day one, a new employee is given access to the various apps and tools they will need to get their job done.
- Six months later, they are promoted, and in the new role they are given access to more tools.
- Two years later, the employee leaves the organization for a new opportunity.

In an organization that does not have Single Sign-On implemented, the employee is likely required to set up multiple sets of credentials on their first day of work — one for each app and tool. Once they are promoted, they are given access to more apps, which means more credentials to set and remember. Finally, when the employee leaves the organization, administrators must track down each account the employee had — i.e., within each and every app and tool — so that they can all be deactivated. Any errors could result in security risks and unnecessary licensing costs.

In an SSO environment, on day one the employee is provided with a set of credentials that are mapped to a user account in a centralized database. The IT administrator grants the user the appropriate permissions within the apps. When the employee is promoted, the IT admin simply updates the user account with new roles and permissions. When the user leaves the organization, the IT administrator simply disables access to all of the organization's resources for that now-departed user's centralized account — a much faster and more reliable approach.

SSO and Identity Federation

Identity Federation and Single Sign-On go hand-in-hand. As outlined above, SSO allows a user to log in once and gain access to multiple disparate apps and services within an organization. Identity Federation offers single access to multiple applications across various organizations.

While Single Sign-On is a feature of Identity Federation, having SSO in place won't necessarily allow for Identity Federation.

Single Sign-On enables a user to log in using an authentication workflow — such as a passkey, social login, or a username and password combination, perhaps also with multi-factor authentication (MFA) — and access multiple applications. This outcome is made possible by a centralized authentication server.

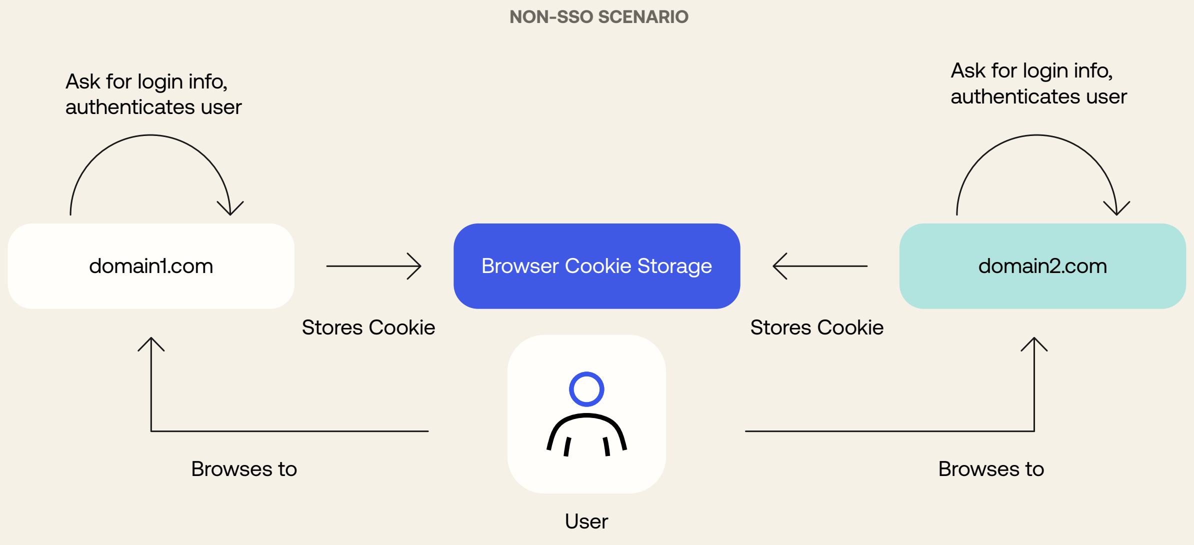
In a non-SSO application, a user will log in and their credentials will be sent to the back-end system — usually the actual application — for verification.

In the SSO use case, the user credentials are sent to a centralized authentication server, and upon verification, this centralized server grants the user the right to access the application they are attempting to log in to.

Without SSO

The diagram below represents a typical non-SSO scenario in which a user is required to authenticate with different credentials on each domain. In this instance, due to the same-origin browser policy — which forbids browsers from sharing cookies between domains — `domain2` would have no way of knowing or accessing data from `domain1`, thus the need to reauthenticate.

Figure 1: In the non-SSO scenario, the user must authenticate separately with both domains



With SSO

In contrast, having a centralized authentication system allows many apps to talk directly to the authentication server to see if a user is authenticated and, if so, to grant them access.

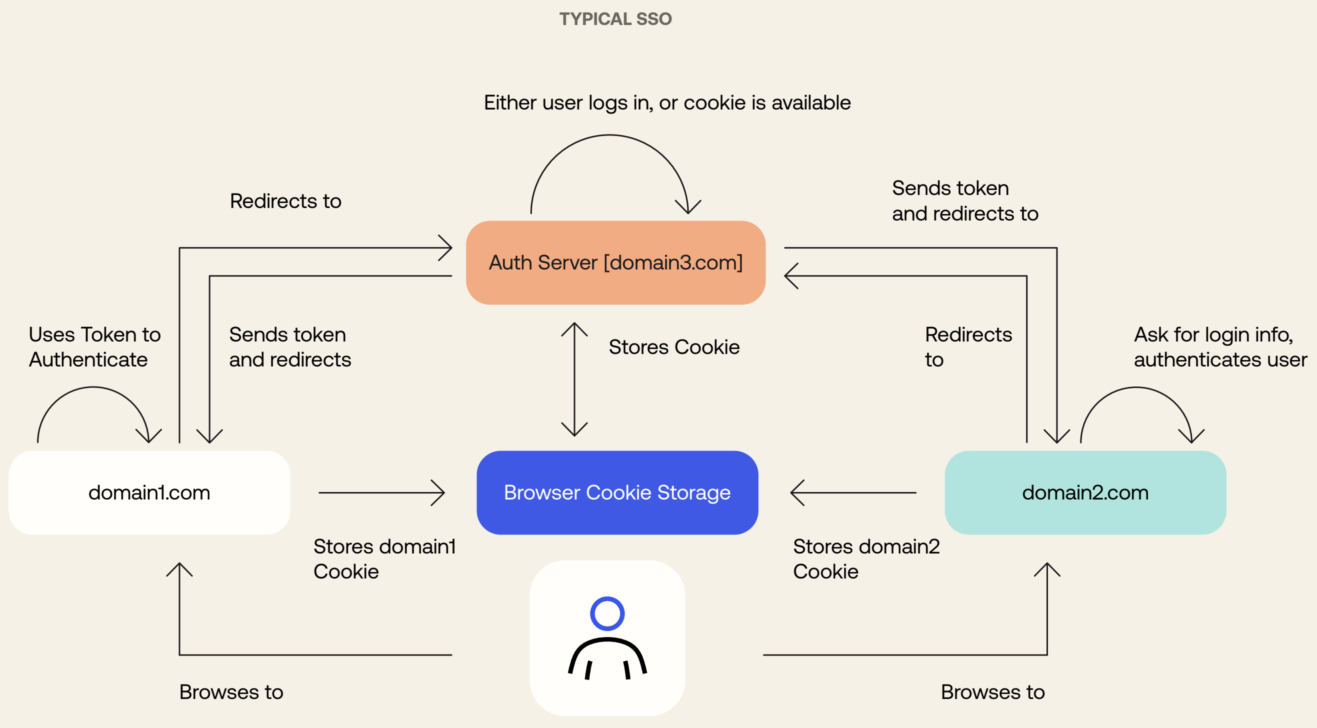
There are multiple ways this can be done:

- In some scenarios, like Google, once a user is logged in to one app they are automatically logged in to all services.
- In other scenarios, even though a user is logged in to one app, they may still be required to provide some information before getting access.

The following diagram shows a typical SSO scenario with two applications. The authorization server lives on `domain3`. `Domain1` and `domain2` interface with `domain3` to both authenticate a user and also to verify that the user is authenticated.

This may seem very complex, but as we dive into the details things will become more clear.

Figure 2: In a typical SSO scenario, apps 'ask' the authorization server if the user is already authenticated


TYPICAL SSO
Either user logs in, or cookie is available
Redirects to
Sends token and redirects to
Auth Server [domain3.com]
Uses Token to Authenticate
Sends token and redirects
Stores Cookie
Redirects to
Ask for login info, authenticates user
domain1.com
Browser Cookie Storage
domain2.com
Stores domain1 Cookie
Stores domain2 Cookie
Browses to
Browses to

Summing up

Single Sign-On assists with three essential functions of modern Identity: authentication, authorization, and Identity management.

Through Identity Federation, an IT administrator can provision and deprovision user accounts once, grant them varying levels of access for each app, and enforce strong authentication all from a single source of truth — all while providing a convenient user experience that keeps team members happy and allows them to do their jobs.

However, implementing SSO is not an easy task. There are many competing standards and protocols, as well as costs associated with implementing an SSO solution.

But because Single Sign-On is so valued by IT administrators, SaaS vendors are able to offer SSO as a premium — i.e., revenue-generating — feature.

Now that we’ve covered what SSO is and why it’s such a necessity, let’s dive more deeply into the various Identity protocols that are used to implement it.

Identity protocols and providers

As we’ve seen, SSO provides many benefits and, accordingly, can be a key feature as you look to sell your application into companies — especially larger enterprises.

Unfortunately, there are many different ways to implement SSO, which can make it challenging for your developers because you’ll likely need to support multiple different protocols.

This section examines the common protocols used in SSO, including their benefits and downsides, and provides examples where these protocols are used. We'll also discuss various Identity Providers and the pros and cons of each.

We'll start with one of the most widely used protocols for managing Federated Identity: SAML.

Security Assertion Markup Language (SAML)

Security Assertion Markup Language, or SAML, is one of the most widely used protocols within Single Sign-On implementations. SAML dates back to

2001, with the last major revision to SAML 2.0 being released in 2005, and for a long time it was the de facto protocol for implementing SSO.

The SAML protocol exchanges authorization and authentication data in XML format. The three roles defined by the protocol are:

- the principal (typically, the user);
- the Identity Provider; and
- the Service Provider.

In this scenario, a user requests a resource from the Service Provider. The Service Provider, before giving the user access to the resources, checks with the Identity Provider to see if the user should have access to this resource. The Identity Provider, acting as the single source of truth, verifies the user's Identity via some form of authentication; if valid, the Identity Provider asserts back to the Service Provider that the user should have access, along with the user Identity.

This exchange is facilitated by assertions. The Service Provider makes assertion requests to the Identity Provider, and the Identity Provider, upon verifying the credentials, returns a response assertion with the user data. An example response assertion can be seen below:

```
<samlp:Response xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol" ID="_621c4c4ae5d60c768cc2" Version="2.0"
IssueInstant="2014-10-14T14:32:17Z" Destination="https://app.auth0.com/tester/samlp">
    <saml:Issuer xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
        urn:matugit.auth0.com
    </saml:Issuer>
    <samlp:Status>
        <samlp:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
    </samlp:Status>
    <saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
        Version="2.0"
        ID="_5VK7LT7FliUkkaQuW6r4brF0DG5E3X76"
        IssueInstant="2014-10-14T14:32:17.251Z">
        <saml:Issuer>
            urn:matugit.auth0.com
        </saml:Issuer>
        <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
            <SignedInfo>
                <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
                <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
                <Reference URI="#_5VK7LT7FliUkkaQuW6r4brF0DG5E3X76">
                    <Transforms>
```

```
                        <Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
                        <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
                    </Transforms>
                    <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                    <DigestValue>
                        ZDkfGO3H1Tu50hawzQVjsACzJwc=
                    </DigestValue>
                </Reference>
            </SignedInfo>
            <SignatureValue>
                …
            </SignatureValue>
            <KeyInfo>
                <X509Data>
                    <X509Certificate>
                     …
                    </X509Certificate>
                </X509Data>
            </KeyInfo>
        </Signature>
        <saml:Subject>
            <saml:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified">
                github|175880
            </saml:NameID>
            <saml:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
                <saml:SubjectConfirmationData
                  NotOnOrAfter="2014-10-14T15:32:17.251Z"
                  Recipient="https://app.auth0.com/tester/samlp"/>
            </saml:SubjectConfirmation>
        </saml:Subject>
        <saml:Conditions
            NotBefore="2014-10-14T14:32:17.251Z"
            NotOnOrAfter="2014-10-14T15:32:17.251Z">
            <saml:AudienceRestriction>
                <saml:Audience>
                    urn:foo
                </saml:Audience>
            </saml:AudienceRestriction>
        </saml:Conditions>
        <saml:AttributeStatement xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
            <saml:Attribute
                Name="http://schemas.xmlsoap.org/ws/2005/05/Identity/claims/nameidentifier">
                <saml:AttributeValue xsi:type="xs:anyType">
                    github|175880
                </saml:AttributeValue>
            </saml:Attribute>
            <saml:Attribute
                Name="http://schemas.xmlsoap.org/ws/2005/05/Identity/claims/emailaddress">
                <saml:AttributeValue xsi:type="xs:anyType">
                    matt@example.com
                </saml:AttributeValue>
            </saml:Attribute>
            <saml:Attribute
```

```
                Name="http://schemas.xmlsoap.org/ws/2005/05/Identity/claims/name">
                <saml:AttributeValue xsi:type="xs:anyType">
                    Matt Mutt
                </saml:AttributeValue>
            </saml:Attribute>
            <saml:Attribute
                Name="http://schemas.xmlsoap.org/ws/2005/05/Identity/claims/upn">
                <saml:AttributeValue xsi:type="xs:anyType">
                    matt@example.com
                </saml:AttributeValue>
            </saml:Attribute>
            <saml:Attribute
                Name="http://schemas.auth0.com/identities/default/access_token">
                <saml:AttributeValue xsi:type="xs:anyType">
                    3a7d0dfeffe12812c37112daa830abef570089b4
                </saml:AttributeValue>
            </saml:Attribute>
            <saml:Attribute
                Name="http://schemas.auth0.com/identities/default/provider">
                <saml:AttributeValue xsi:type="xs:anyType">
                    github
                </saml:AttributeValue>
            </saml:Attribute>
            <saml:Attribute
                Name="http://schemas.auth0.com/identities/default/connection">
                <saml:AttributeValue xsi:type="xs:anyType">
                    github
                </saml:AttributeValue>
            </saml:Attribute>
            <saml:Attribute
                Name="http://schemas.auth0.com/identities/default/isSocial">
                <saml:AttributeValue xsi:type="xs:anyType">
                    true
                </saml:AttributeValue>
            </saml:Attribute>
        </saml:AttributeStatement>
        <saml:AuthnStatement AuthnInstant="2014-10-14T14:32:17.251Z">
            <saml:AuthnContext>
                <saml:AuthnContextClassRef>
                    urn:oasis:names:tc:SAML:2.0:ac:classes:unspecified
                </saml:AuthnContextClassRef>
            </saml:AuthnContext>
        </saml:AuthnStatement>
    </saml:Assertion>
</samlp:Response>
```

These XML-based requests and responses are self-encompassing and independent of the protocol used to transfer them — so developers can send these requests in the body of a POST request, for example, or wrap them in a SOAP binding. This flexibility means that SAML can be introduced in many different environments and it will just work.

One of the major benefits of SAML is that the protocol is only concerned with handling the exchange between the Identity Provider and Service Provider. The authentication aspect is not defined in the protocol, so developers are free to choose how users authenticate. For example, if you have an existing database of users, you can have them validate against that database rather than rebuilding your Identity management system from scratch. You can also enable enhanced security features like multi-factor authentication without impacting your SAML implementation.

With SAML, you are not limited to the number of Identity or Service Providers. One Identity Provider can interact with many Service Providers and one Service Provider can interact with many Identity Providers. This capability ensures that your organization will be able to interact with not just your own internal applications, but also third-party applications that you may be using.

WS-Federation

WS-Federation, or WS-Fed, is another protocol commonly used in Single Sign-On implementations. For the most part, it is very similar to SAML. WS-Fed comes from the WS-* series of services, falling under the WS-Security branch. WS or Web Services aim to provide a standardized way of handling online communications with varying degrees of success and maturity.

If your organization is already using WS-* services for security and trust, WS-Fed will allow you to extend these capabilities to Single Sign-On scenarios. WS-Fed is built on top of WS-Trust which uses the Security Token Services (STS) model to facilitate the transfer of claims data between services.

Like SAML, WS-Federation is built around the concept of an Identity Provider and Service Provider, in which a Service Provider requests Identity information from an Identity Provider, while the Identity Provider is tasked with verifying incoming requests have the correct set of credentials.

The WS-Federation protocol is also XML-based and looks very similar to that of SAML. An example of federation metadata can be seen below:

```
<fed:FederationMetadata>
        <fed:Federation>
                <fed:TokenIssuerName>
```

```
                    http://www.auth0.com
                </fed:TokenIssuerName>
                <fed:TokenIssuerEndpoint>
                    <wsa:Address>
                        http://www.auth0.com/STS
                    </wsa:Address>
                </fed:TokenIssuerEndpoint>
                <fed:TokenTypesOffered>
                    ...
                </fed:TokenTypesOffered>
                <fed:SingleSignOutNotificationEndpoint>
                    <wsa:Address>
                        http://www.auth0.com/SignOut
                    </wsa:Address>
                </fed:SingleSignOutNotificationEndpoint>
                <fed:UriNamedClaimTypesOffered>
                    <fed:ClaimType Uri="http://schemas.xmlsoap.org/claims/EmailAddress">
                        <fed:DisplayName>
                            Email
                        </fed:DisplayName>
                    </fed:ClaimType>
                    <fed:ClaimType Uri="http://schemas.xmlsoap.org/claims/UPN">
                        <fed:DisplayName>
                            Name
                        </fed:DisplayName>
                    </fed:ClaimType>
                </fed:UriNamedClaimTypesOffered>
                <fed:TokenSigningKeyInfo>
                    <wsse:SecurityTokenReference>
                        <ds:X509Data>
                            <ds:X509Certificate>
                            ...
                            </ds:X509Certificate>
                        </ds:X509Data>
                    </wsse:SecurityTokenReference>
                </fed:TokenSigningKeyInfo>
        </fed:Federation>
</fed:FederationMetadata>
```

WS-Federation shares many benefits already discussed in the SAML protocol. This protocol was developed to extend the capabilities of the existing WS-Security model. WS-Federation is commonly found in large enterprise companies, especially those embedded in the Microsoft ecosystem.

OpenID Connect / OAuth

OpenID Connect (OIDC) is an authentication protocol, based on the OAuth 2.0 family of specifications. It handles authentication via JSON Web

Tokens (JWTs) delivered via the OAuth 2.0 protocol. Compared to SAML and WS-Federation, OpenID Connect is a fairly recent protocol, with version 1.0 of the framework being adopted in 2014. OpenID Connect is typically used by social networks that allow developers to use them as Identity Providers.

While OAuth 2.0 is concerned with resource access and sharing, OIDC is all about user authentication. Like the previous protocols, OIDC facilitates access through a centralized Identity Provider. A user wishing to gain access to an application utilizing OIDC, will first be redirected to the OIDC provider, authenticate, then the Identity Provider will return the user's Identity to the application.

The flow for OpenID Connect is different from that of SAML and WS-Federation. With OpenID Connect, an application sends a request to an Identity Provider. The Identity Provider verifies the user and upon successful verification, prompts the user to grant data access to the application that initiated the request. Once a user agrees to share the data, the Identity Provider generates a token, called an ID token and returns it to the application. This token contains user Identity information and the application consumes this token and grants the user access.

The ID token only proves the user has been authenticated and provides Identity information. If the application also needs authorization information, a second token is sent to the application: the access token. The access token represents the permissions granted by the user to the application. Its format is not necessarily JWT, although this format is commonly used.

JSON Web Tokens are widely used in OAuth and OIDC. A JSON Web Token or JWT consists of three parts: the header, payload, and signature. Each JWT is signed by an algorithm such as HMAC SHA256 and can only be verified by the correct key. While token verification can only be done with the right credentials, JWTs are only encoded, not encrypted. This means that they can be decoded and their contents read by anybody. For this reason, it is recommended that sensitive information never be stored in these tokens. To learn more about JSON Web Tokens, be sure to check out the introduction at https://jwt.io/introduction.

For the purposes of OIDC, each token contains Identity data for the user in the form of claims. The OIDC spec defines a series of reserved claims, public claims and private claims. Reserved claims are typically reserved for metadata, for example `iss` is for the issuer of the token. Public claims are agreed upon in the IANA JSON Web Token Registry, where claims like `given_name` are defined for the user's first name. This is to help with interoperability. Finally private claims are those that an application or Identity Provider defines themselves. These can be named anything and represent any type of data the Identity Provider wants to represent.

The format of JSON Web Tokens is that of:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiYWRtaW4iOnRydWV9.
TJVA95OrM7E2cBab30RMHrHDcEfxjoYZgeFONFh7HgQ
```

When decoded the claims represented in the payload of the JWT are:

```
{
        "sub": "1234567890",
        "name": "John Doe",
        "admin": true
}
```

OpenID Connect is a fairly modern approach to Single Sign-On but is widely adopted in the consumer market. If you've ever logged into a web or mobile application with your Google or Facebook account, you've used OpenID Connect. OIDC is not as common in the enterprise environment as it is with consumer facing applications.

Lightweight Directory Access Protocol (LDAP)

The Lightweight Directory Access Protocol (LDAP) is an application protocol, used for accessing and maintaining distributed directory information services over an Internet Protocol (IP) network.

The LDAP protocol allows access to a centralized directory of credentials which can be shared amongst multiple applications. The LDAP protocol is the oldest one we'll look at in this guide, dating back to 1996.

LDAP works by enabling access to an existing directory of users. LDAP is the basis of Microsoft’s Active Directory and allows administrators to have a centralized location for user Identity. Application clients send requests to the LDAP server and in return the server responds with information about user Identity. There are a number of different actions a client can request such as authenticating a user, updating user information, deleting users, and more.

Information exchange over LDAP is done over LDAP Data Interchange Format (LDIF), which looks looks like this:

```
dn: cn=John Doe,dc=example,dc=com
cn: John Doe
givenName: John
sn: Doe
mail: john@example.com
manager: cn=Jane Doe,dc=example,dc=com
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: person
objectClass: top
```

Companies may use the LDAP protocol to provide SSO services to their employees within an intranet. LDAP-based SSO is not common outside of company intranets.

Identity Providers

The Identity protocols dictate how the flow of Single Sign-On works among the user, the application, and the Identity Provider. Identity Providers are concerned with actually storing and managing user identities. In this section, we are going to look at the different ways of storing user Identity.

Local database
A common way of managing Identity is through a local database which is only concerned with users and their roles. Organizations requiring full control over their data will often opt for this option for their Identity Provider. Traditionally, users typically log in to the database via username and password authentication — although organizations are gradually implementing stronger authentication and other enhanced security measures.

Microsoft: Active Directory, ADFS, Entra
Microsoft is no stranger to Federated Identity and has a long history of offering various products and solutions for managing user Identity.

Active Directory is one of the most popular Identity Providers in the Microsoft Enterprise space. Active Directory worked with Windows Server technologies to provide Single Sign-On functionality to not just web applications but the entire Windows ecosystem. In recent years, Microsoft has begun offering their various online services like Microsoft Entra (formerly known as Microsoft Azure Active Directory or Azure AD), bringing their tried and true Identity Provider to the cloud.

Social providers
Social providers can often make for great Identity Providers. Companies like Facebook and Google typically make use of the OpenID Connect protocol for managing user Identity.

The benefit of using a large social provider as an Identity Provider is that these organizations typically have some of the best security standards for user accounts in the world. For organizations like Facebook, that hold billions of identities, top-notch security is a must — to the point where many of these organizations are writing the rulebooks for managing modern Identity.

Using social providers as your Identity Provider is not without risk, though. Social providers typically have all the control and can revoke access, change their API, or stop providing Identity services altogether in the blink of an eye.

Auth0 by Okta
Okta is a leader in Identity and Access Management, and Auth0 by Okta is the solution that can fulfill your organization's SSO needs. Beyond providing you with Single Sign-On, it offers additional services to make Identity management simple for your organization.

Organizations both large and small are demanding Single Sign-On and it has become a requirement for companies wishing to compete in today's competitive landscape. SSO should be a check-the-box item for your organization during due diligence rather than a capability your organization must demonstrate to a potential customer.

In the build vs buy debate, the optimal approach is to build what makes your product unique and buy everything else. Identity management is a prime candidate that can be delegated to a specialized vendor.

Use cases

It’s hard to overstate the benefits of SSO, but it's also important to ensure that SSO is a right fit for your organization. So far we've looked at Single Sign-On through the value proposition and technological lenses, but now we’ll look at it through the lens of real-world use cases.

SSO for your organization

Whether you’re a small startup or large enterprise, Single Sign-On enables the consolidation of user Identity and management. Think of how many different applications you use every day: email, an issue tracker, file hosting, CRM software, and many more. If you require a different set of credentials for each one of these apps, then you likely have a very fragmented Identity system. Many software vendors offer SSO integrations, but they do come at a higher price.

If you wish to get your Identity management consolidated under one roof, Single Sign-On can help. Consolidating Identity in an organization through SSO will require a centralized Identity Provider, and depending on your existing infrastructure you are likely going to want to use SAML or WS-Federation.

SSO for your applications

If your organization develops applications, Single Sign-On can be both a differentiator and a requirement. Adding this feature can seem like a daunting task, but companies like Okta can help make the integration simple. Let's examine how Single Sign-On can benefit your business.

Business-to-Consumer (B2C) applications
Building applications for consumers, such as e-commerce or media applications, means getting the user experience right. Consumers have an abundance of choice and a bad first impression is enough to get them to leave and never come back.

The login screen is typically the first experience users have with your application — and offering a Single Sign-On solution here can mean the difference between a conversion and an abandonment.

Offering Single Sign-On through social logins is a great way to get new signups and additional data from users. The experience for consumers is convenient and familiar, as they don’t have to remember another set of credentials — they just log in with their existing social media account.

This ease of use makes social logins ideal for B2C applications, and OpenID Connect-based SSO is the implementation that will achieve this goal.

The challenge with providing Single Sign-On solutions for the B2C market is choosing which Identity Providers to connect to. Large IdPs like Facebook, Google, and Twitter are the obvious choices, but not the only ones. For example, offering GitHub makes sense if your application is focused on developers. Supporting multiple IdPs and giving consumers choice is common, but the task can be time- and cost-intensive. Identity Providers may change their APIs or policies requiring your organization to move fast to keep up with the changes.

For your own employees, a traditional SAML-based SSO solution may also be applicable. This would allow your employees to access the B2C application without having to create a separate account. This would make managing internal Identity a lot easier for your IT team.

Business-to-Business (B2B) applications
The demand for Single Sign-On in the B2B space is on the rise, as security leaders and IT teams aim to strengthen safeguards and enable productivity.

For SaaS providers, this demand represents both a general business opportunity as well as increased revenue potential.

For many organizations, maintaining the ownership of their user identities is a requirement and the only way to achieve this result is through Single Sign-On. No matter how good your app is at its main functions, if it lacks support for SSO then it may be a non-starter for many potential customers.

For B2B software, your organization will typically need to support SAML, LDAP or WS Federation SSO. SAML-based SSO is most commonly found in B2B SaaS, so that is a great starting point, but the more protocols your organization can support the better.

Business-to-Enterprise (B2E) applications
Single Sign-On is pretty much a requirement when selling to the enterprise. Large enterprises demand governance over their users, and the only way to ensure compliance is through SSO.

SAML, WS-Federation, and LDAP-based SSO are the standards here — although OAuth and OpenID Connect are gaining ground.

Supporting many different variations of SSO — supporting many different protocols and IdPs — is essential, as large enterprises typically run many custom applications, with the result that each enterprise is likely to have unique requirements.

To build or to buy?

No use case discussion would be complete without exploring how to achieve the goal. Do you build SSO capabilities internally and manage the implementation yourself, or is it better to purchase SaaS that provides SSO solutions that can be integrated into your products?

Building your own solution, whether it be a blog, framework, or Identity can allow you to build to your exact specifications, with complete control over all of the data as well as how and where it is stored. Plus, reducing dependencies on external vendors can help to inoculate your app against service outages. And — at least theoretically, and at least in the beginning — you can manage costs by only building what you need.

However, as many developers have discovered, while Identity can seem conceptually straightforward at first glance:

- **Getting Identity right is extremely challenging**, even for experienced engineering teams; and
- **Identity is constantly changing** — new standards get defined, bugs get identified, and needs of businesses evolve — which means that, in practice, building your own implementation is only the first step of an ongoing, extensive, distracting, and ultimately costly endeavor.

Alternatively — as we explain in detail in Build or buy? Why developers from early stage startups to the most mature enterprises choose a ready-made Customer Identity solution — integrating a ready-built solution…

- Frees developers to focus on core product needs
- Shortens time-to-market (TTM)
- Enables you to keep pace with changes beyond your control
- Increases your agility
- Is your best defense against Identity attacks before, at, and after login
- Helps you meet regulatory, framework, and standards obligations

- Positions you to pursue and win enterprise customers
- Provides you with access to valuable developer resources and operational support
- Is — ultimately — the most cost-effective approach in all but a few very limited circumstances

Alright, now it's time to get hands-on with SSO!

Implementing Single Sign-On

So far we've covered many topics including the what, why, where, and when to use Single Sign-On. In this section, we'll turn our attention to the how — specifically, how to:

- Implement SAML-based SSO
- Use Auth0 as the Identity Provider
- Authenticate various clients via SSO

We'll implement SAML-based SSO in our applications with Auth0 serving as our Identity Provider. We've chosen this implementation because SAML is an open standard and Auth0 provides an easy-to-use Identity framework with many features we've already examined.

The starting scenario

Let's imagine ourselves in a company that intends to implement SSO both to improve the employee experience and to simplify and strengthen user management. The company uses a few common tools for communication and collaboration: Google Workspace and Asana. It also has a custom application developed in-house for some core business management functions.

The company does not currently use SSO, so each employee is forced to use a separate set of credentials for each application. That's three logins to remember, three sets of credentials to keep safe, and three identities for the IT administrator to take care of.

The company is also looking to build an analytics app to better understand how customers are using their product. So that's another set of credentials to deal with.

Luckily, the company is in the exploratory stages of Identity and authentication, so you suggest they explore a Single Sign-On solution to consolidate the user accounts across all these applications. Your idea is approved, but you've been tasked with implementing it.

It would seem that no good deed goes unpunished.

The initial steps

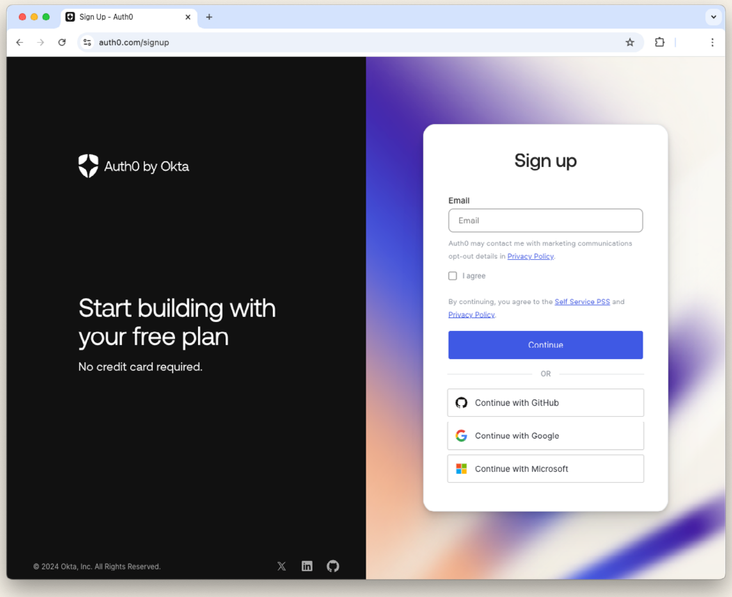
Now that you’ve been tasked with implementing SSO in your company, let's examine the first steps.

Deciding on the Identity Provider

The first step to setting up Single Sign-On is to decide on an Identity Provider. Recall from previous sections that the Identity Provider is the source of truth for Identity in an SSO system, with all accounts managed by the IdP.

There are many options and considerations for deciding on an IdP. If you visit this Wikipedia page, you can see a large number of different vendors that provide IdP services. Some work with proprietary protocols like Active Directory, while others provide open source alternatives through SAML, OpenID Connect, and so on.

As our Identity Provider, we'll use Auth0, and we'll build our SSO using SAML. To get started, sign up for a free Auth0 account at https://auth0.com/signup.



Start building with your free plan

No credit card required.

Auth0 may contact me with marketing communications opt-out details in Privacy Policy.

By continuing, you agree to the Self Service PSS and Privacy Policy.

© 2024 Okta, Inc. All Rights Reserved.

Once you have your Auth0 account, you are ready to implement SSO for your company.

Defining the high-level architecture
In the final scenario, Auth0 will be the IdP for all applications involved: Google Workspace, Asana, and the custom application. Users will have only one set of credentials to access all three applications.

To set up SAML-based SSO with the two third-party applications, you need to instruct Auth0 how to communicate with them as an IdP.

You need to do the same for the custom application, but in this case, Auth0 also acts as a Service Provider because it connects the result of the authentication process to your application. In other words, in the case of your custom application, Auth0 plays both the role of IdP and Service Provider. In the following, we will treat the two cases separately.

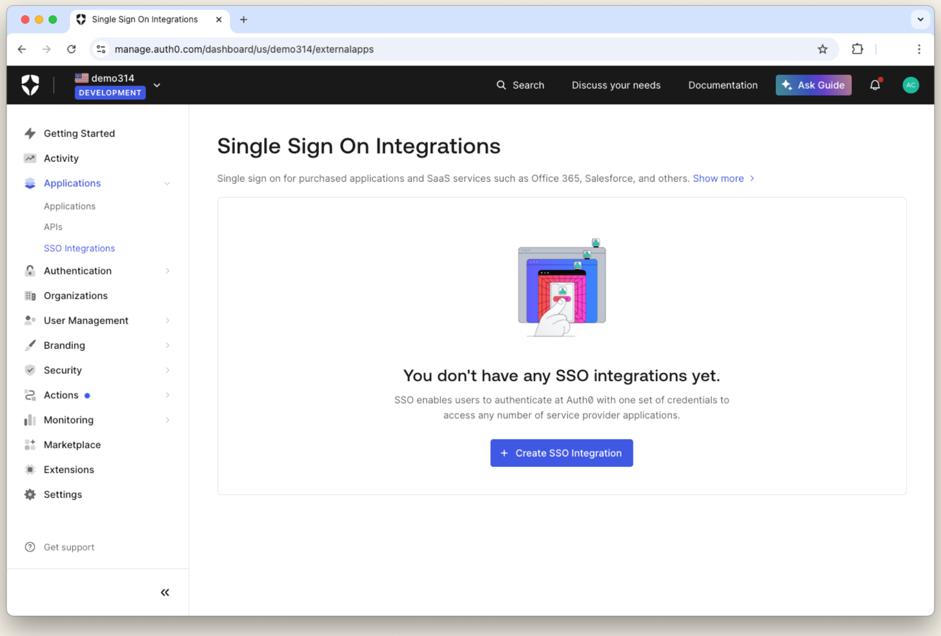
Auth0 as the SAML Identity Provider

First, you'll implement SSO for the two third-party applications. In this case, Auth0 only acts as the IdP. Let's start by configuring Auth0 as an IdP for Google Workspace.

Configure the SSO integration
You will configure Auth0 as the IdP for Google Workspace in two steps:

1. You will create an SSO integration on the Auth0 side; next
2. You will provide Google Workspace with some configuration parameters from Auth0.

Let's start the first step. Access your Auth0 dashboard and go to Applications > SSO Integrations:



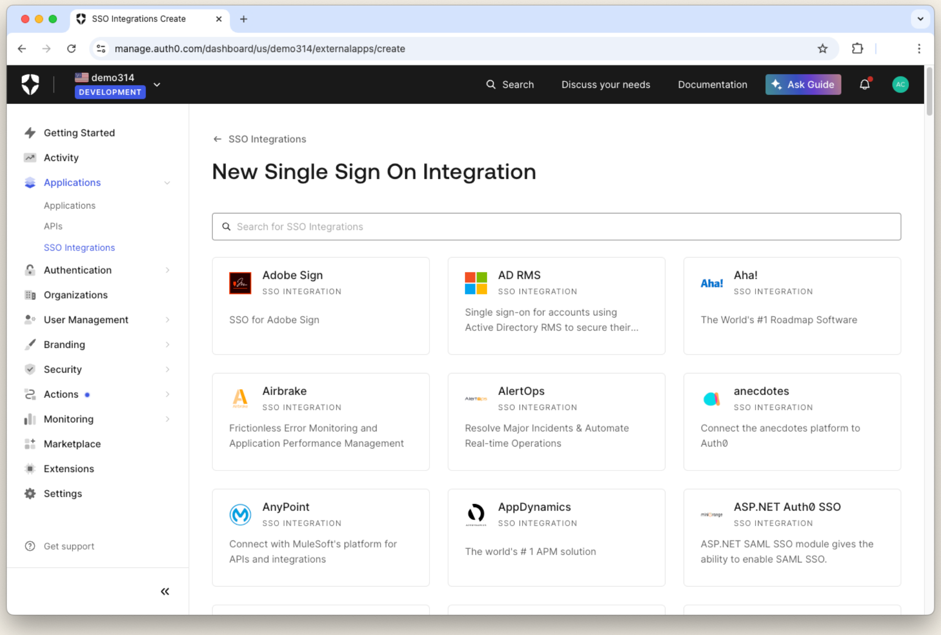
Single Sign On Integrations

Single sign on for purchased applications and SaaS services such as Office 365, Salesforce, and others. Show more >

You don't have any SSO integrations yet.

SSO enables users to authenticate at Auth0 with one set of credentials to access any number of service provider applications.

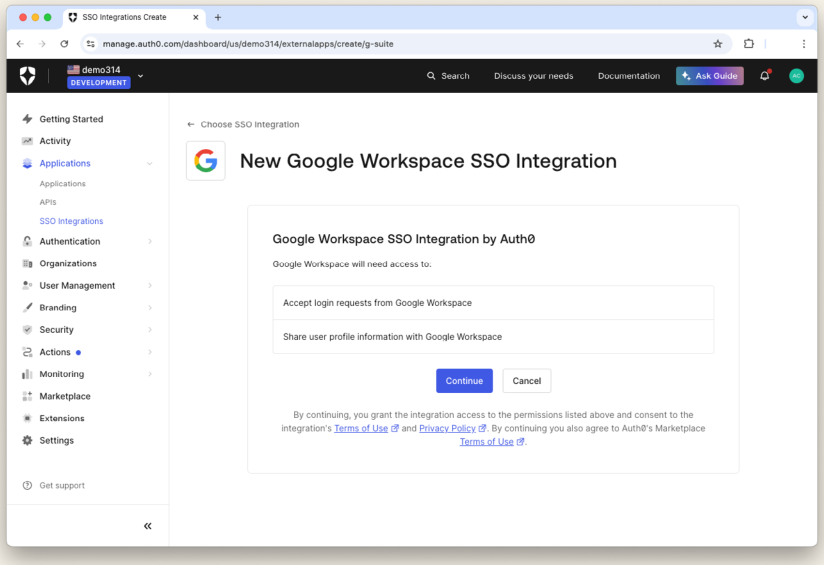
Now click the **Create SSO Integration** button. A list of SSO integrations will show, as in the following picture:



New Single Sign On Integration

Adobe Sign SSO INTEGRATION SSO for Adobe Sign	AD RMS SSO INTEGRATION Single sign-on for accounts using Active Directory RMS to secure their...	Aha! SSO INTEGRATION The World's #1 Roadmap Software
Airbrake SSO INTEGRATION Frictionless Error Monitoring and Application Performance Management	AlertOps SSO INTEGRATION Resolve Major Incidents & Automate Real-time Operations	anecdotes SSO INTEGRATION Connect the anecdotes platform to Auth0
AnyPoint SSO INTEGRATION Connect with MuleSoft's platform for APIs and integrations	AppDynamics SSO INTEGRATION The world's # 1 APM solution	ASP.NET Auth0 SSO SSO INTEGRATION ASP.NET SAML SSO module gives the ability to enable SAML SSO.

Filter the list by looking for Google Workspace and select the resulting item. You will see a page telling you what this integration will do:



← Choose SSO Integration

New Google Workspace SSO Integration

Google Workspace SSO Integration by Auth0

Google Workspace will need access to:

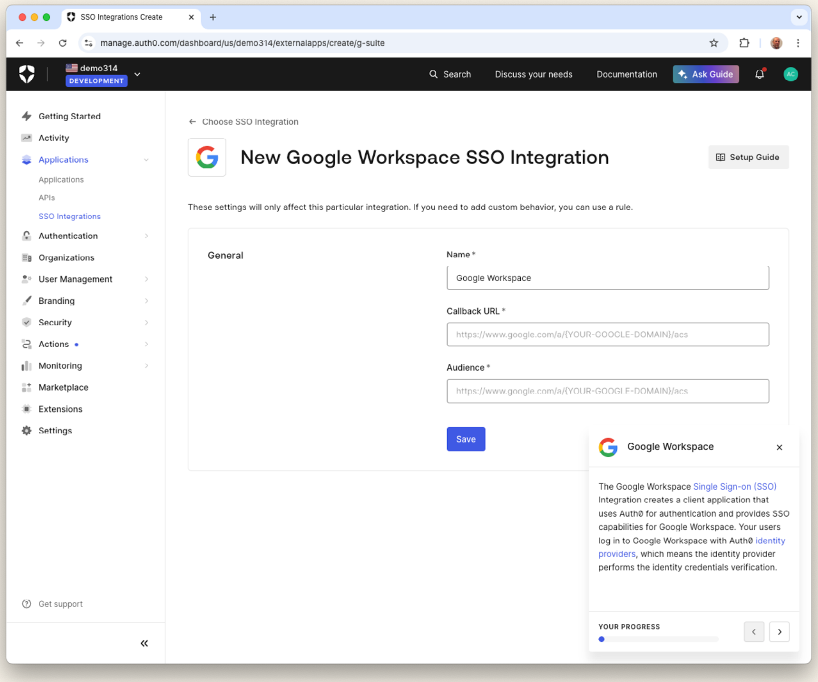
Accept login requests from Google Workspace

Share user profile information with Google Workspace

Continue Cancel

By continuing, you grant the integration access to the permissions listed above and consent to the integration's Terms of Use and Privacy Policy. By continuing you also agree to Auth0's Marketplace Terms of Use.

Click Continue and you will land on the following screen:



← Choose SSO Integration

New Google Workspace SSO Integration

Setup Guide

These settings will only affect this particular integration. If you need to add custom behavior, you can use a rule.

General

Name *
Google Workspace

Callback URL *
https://www.google.com/a/{YOUR-GOOGLE-DOMAIN}/acs

Audience *
https://www.google.com/a/{YOUR-GOOGLE-DOMAIN}/acs

Save

Google Workspace

The Google Workspace Single Sign-on (SSO) Integration creates a client application that uses Auth0 for authentication and provides SSO capabilities for Google Workspace. Your users log in to Google Workspace with Auth0 identity providers, which means the identity provider performs the identity credentials verification.

YOUR PROGRESS

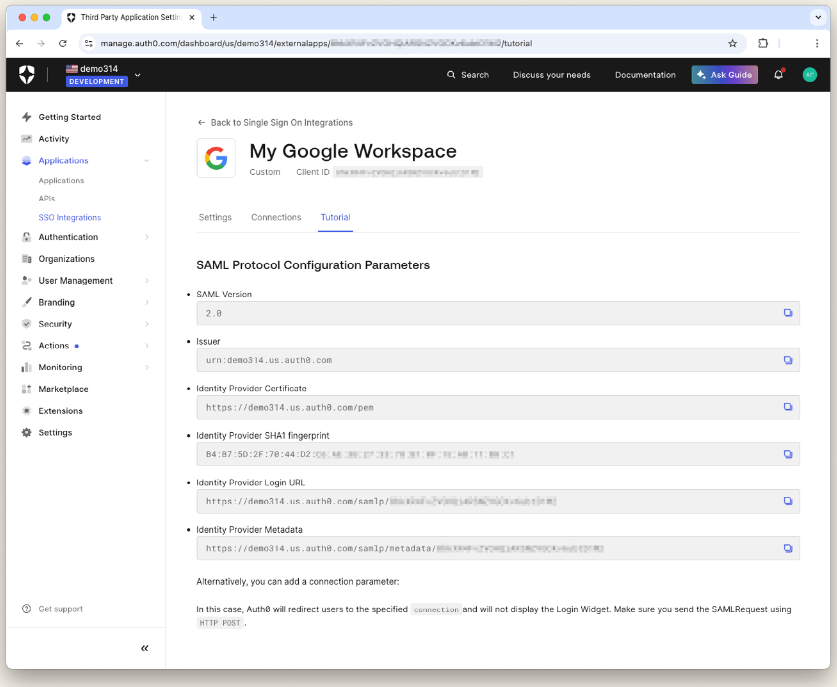
You need to provide the configuration details of your company's Google Workspace instance. Auth0 needs this data to communicate the result of user authentication to it.

Specifically, you need to add:

- A name that identifies the SSO integration you are creating;
- The callback URL (i.e., the URL where Auth0 will redirect users after authentication), also known as the Assertion Consumer Service (ACS) URL. In the case of Google, this URL has the form of `https://www.google.com/a/{YOUR-DOMAIN}/acs` , where `{YOUR-DOMAIN}` is a placeholder for your company's Google Workspace domain.
- The audience (i.e., the URI that identifies the intended Service Provider you are connecting to). In the case of Google, it has the same form as the callback URL.

Check out Google's documentation to learn more about the configuration parameters on Google's side.

Once you provide the required parameters, click the Save button and you'll see a set of parameters related to Auth0 as your SAML IdP:



My Google Workspace

Custom Client ID [redacted]

SAML Protocol Configuration Parameters

- SAML Version
  `2.0`
- Issuer
  `urn:demo314.us.auth0.com`
- Identity Provider Certificate
  `https://demo314.us.auth0.com/pem`
- Identity Provider SHA1 fingerprint
  `B4:B7:5D:2F:70:44:D2:`[redacted]
- Identity Provider Login URL
  `https://demo314.us.auth0.com/samlp/`[redacted]
- Identity Provider Metadata
  `https://demo314.us.auth0.com/samlp/metadata/`[redacted]

Alternatively, you can add a connection parameter:

In this case, Auth0 will redirect users to the specified `connection` and will not display the Login Widget. Make sure you send the SAMLRequest using `HTTP POST`.

For the second step, make sure you have administrator permissions and follow Google's instructions to configure SSO for your company's Google Workspace instance.

Test the SSO integration
Now that you have configured Auth0 as your SAML IdP and Google Workspace as your SAML Service Provider, let's ensure everything works as expected.

As a first step, make sure you have at least one user created in your current Auth0 tenant. These users will be enabled to access Google Workspace through Auth0. If you don't have any, create one following these steps.

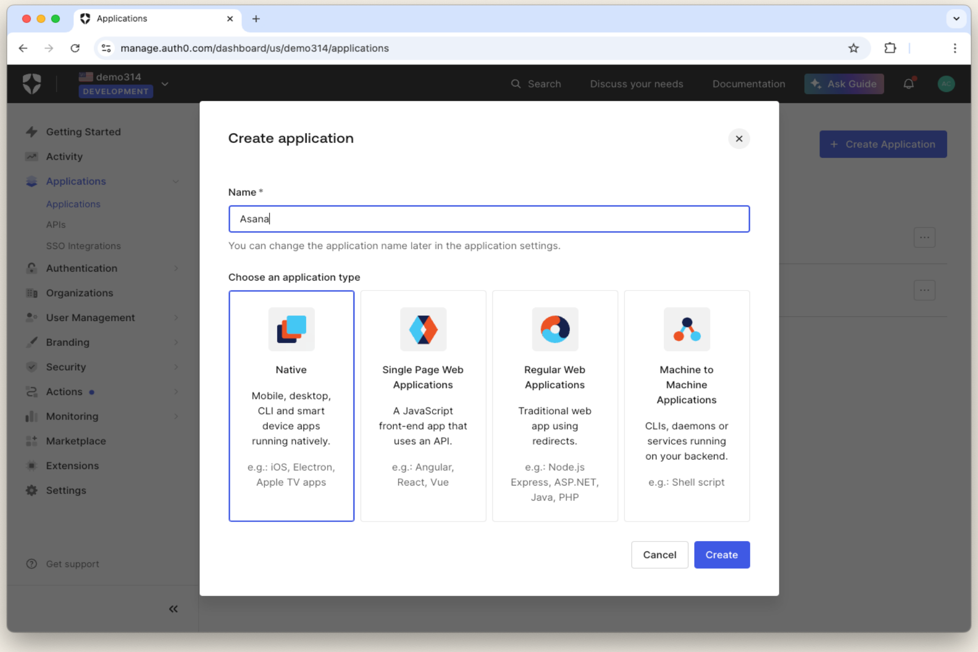
Then, clear your browser history and cookies — this ensures that you are not using an existing authenticated session.

Now, access one of the Google Workspace applications, such as GMail. You should be redirected to the Auth0 login page. After you enter your user's credentials, you should go back to GMail.

Awesome! Now Auth0 is the Identity Provider for your Google Workspace applications.

Custom SSO integration
In case you need to configure a Service Provider for which there is no SSO integration available in the Auth0 dashboard, you can configure it manually. For example, at the time of writing, there is no SSO integration for Asana. You can configure Auth0 as your SAML IdP in two steps as in the previous case, but with a slightly different approach.

For the first step, go to your Auth0 dashboard and click the Applications > Applications menu. Here, create a new application by giving it a friendly name (such as Asana, in our case):



Create application

Name *

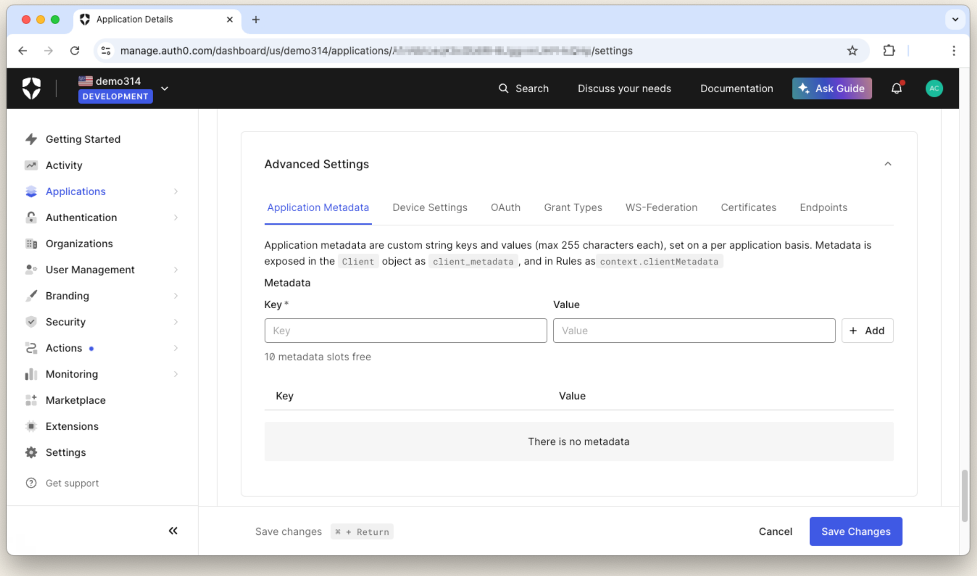
Asana

You can change the application name later in the application settings.

Choose an application type

Native	Single Page Web Applications	Regular Web Applications	Machine to Machine Applications
Mobile, desktop, CLI and smart device apps running natively.	A JavaScript front-end app that uses an API.	Traditional web app using redirects.	CLIs, daemons or services running on your backend.
e.g.: iOS, Electron, Apple TV apps	e.g.: Angular, React, Vue	e.g.: Node.js Express, ASP.NET, Java, PHP	e.g.: Shell script

Then, go to the **Settings** tab and scroll down to the **Advanced Settings** section and click on it. You will see the following screen:



Advanced Settings

Application metadata are custom string keys and values (max 255 characters each), set on a per application basis. Metadata is exposed in the `Client` object as `client_metadata`, and in Rules as `context.clientMetadata`

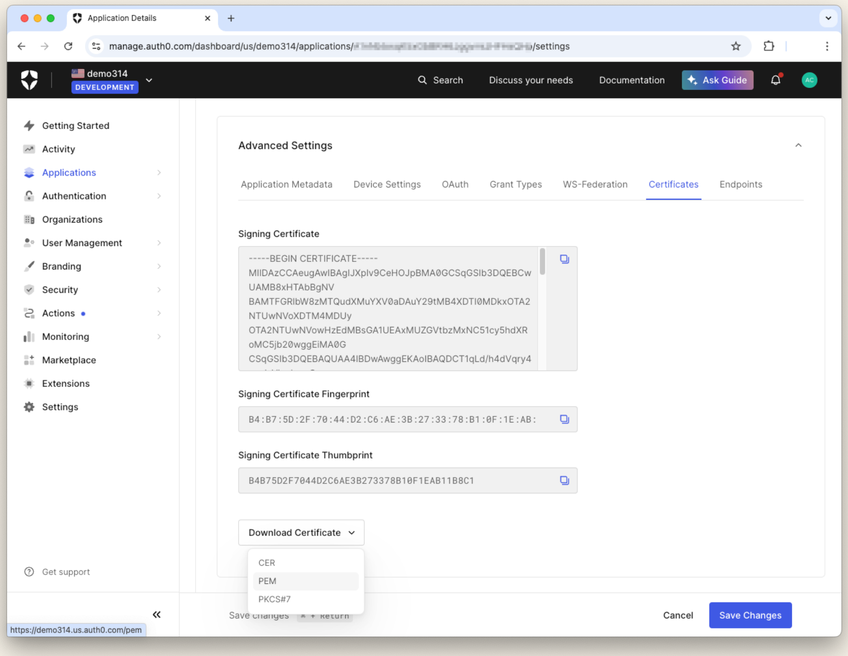
Metadata

Key * Value

10 metadata slots free

Key	Value
There is no metadata	

Go to the **Certificates** tab and click the **Download Certificate** button at the bottom of the section:



Signing Certificate

```
-----BEGIN CERTIFICATE-----
MIIDAzCCAeugAwIBAgIJXplv9CeHOJpBMA0GCSqGSIb3DQEBCw
UAMB8xHTAbBgNV
BAMTFGRlbW8zMTQudXMuYXV0aDAuY29tMB4XDTI0MDkxOTA2
NTUwNVoXDTM4MDUy
OTA2NTUwNVowHzEdMBsGA1UEAxMUZGVtbzMxNC51cy5hdXR
oMC5jb20wggEiMA0G
CSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDCT1qLd/h4dVqry4
```

Signing Certificate Fingerprint

```
B4:B7:5D:2F:70:44:D2:C6:AE:3B:27:33:78:B1:0F:1E:AB:
```

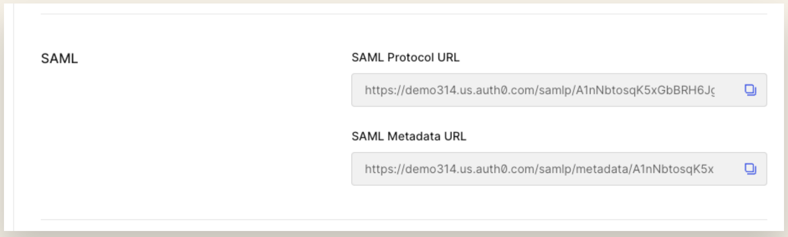
Signing Certificate Thumbprint

```
B4B75D2F7044D2C6AE3B273378B10F1EAB11B8C1
```

You can download the certificate in several formats. The PEM format is pretty common, but check your Service Provider documentation to learn what format it accepts.

You will get the certificate in a file named as your tenant. For example, if you choose PEM as the certificate format for your tenant named *foo*, you will get a file named `foo.pem`. You will use this file to configure your Asana instance as a Service Provider.

Go to the **Endpoints** tab of the **Advanced Settings** section and locate the **SAML** section. You should see two URLs similar to the ones shown below:



SAML Protocol URL

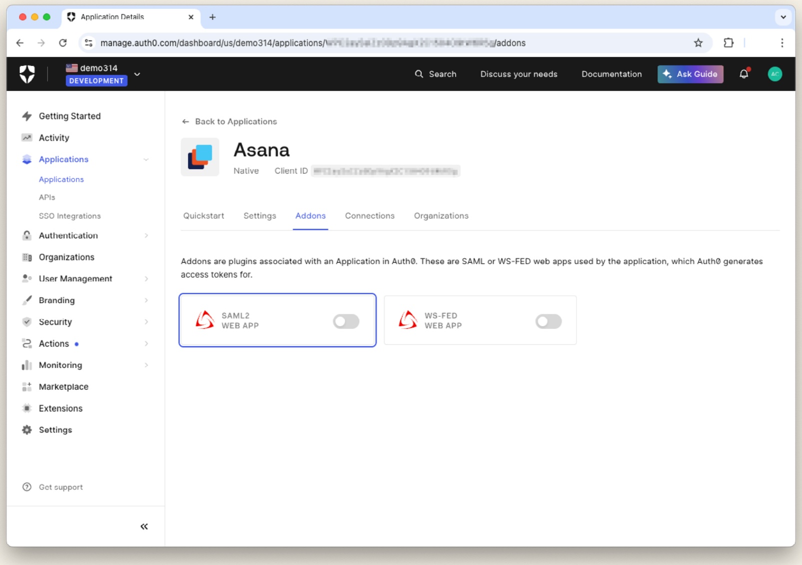
https://demo314.us.auth0.com/samlp/A1nNbtosqK5xGbBRH6J

SAML Metadata URL

https://demo314.us.auth0.com/samlp/metadata/A1nNbtosqK5x

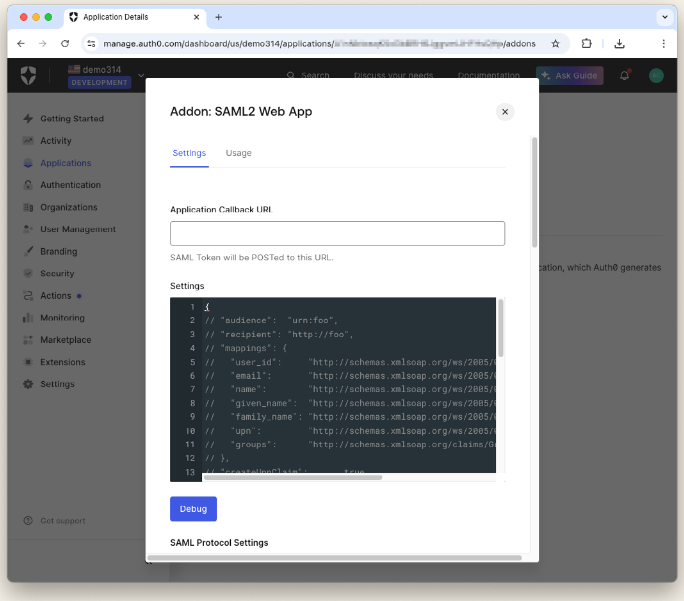
Take note of these URLs — you will need them to configure your Asana instance.

Now go to the top of the page and select the **Addons** tab of the Asana application configuration page. Select the **SAML2 Web App** toggle:



Addons are plugins associated with an Application in Auth0. These are SAML or WS-FED web apps used by the application, which Auth0 generates access tokens for.

In the popup window that appears, select the **Settings** tab:



SAML Token will be POSTed to this URL.

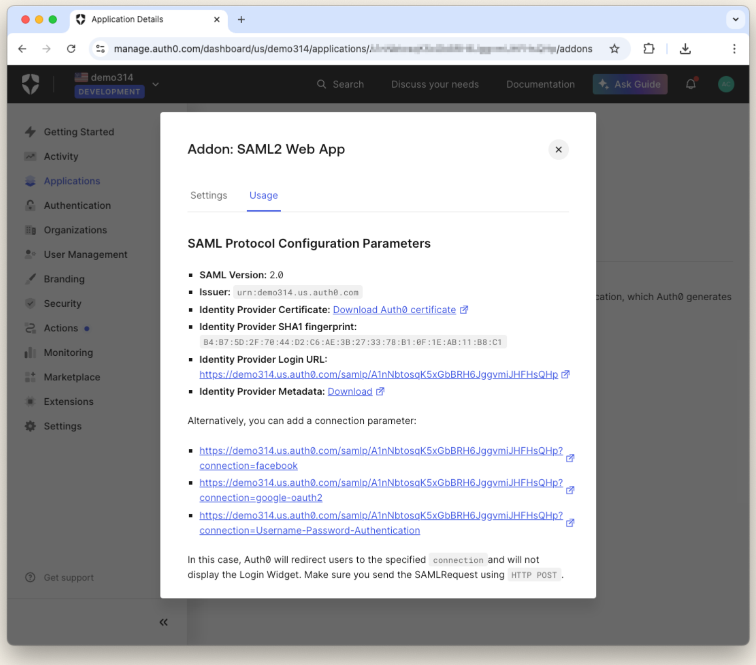
```
{
// "audience":  "urn:foo",
// "recipient": "http://foo",
// "mappings": {
//   "user_id":     "http://schemas.xmlsoap.org/ws/2005/
//   "email":       "http://schemas.xmlsoap.org/ws/2005/
//   "name":        "http://schemas.xmlsoap.org/ws/2005/
//   "given_name":  "http://schemas.xmlsoap.org/ws/2005/
//   "family_name": "http://schemas.xmlsoap.org/ws/2005/
//   "upn":         "http://schemas.xmlsoap.org/ws/2005/
//   "groups":      "http://schemas.xmlsoap.org/claims/G
// },
// "createUpnClaim":       true
```

Enter the **Assertion Consumer Service (ACS) URL** in the **Application Callback URL** field. This is the Service Provider endpoint where SAML assertions will be sent after Auth0 has authenticated the user. At the time of writing, the URL for Asana is https://app.asana.com/-/saml/consume.

Check out Asana documentation to learn more about the configuration parameters on Asana's side.

Scroll down to the bottom of the tab and click the **Enable** button to save your settings.

Now switch to the **Usage** tab of the **SAML Addon** to get the information you need to configure Asana as your Service Provider:



Addon: SAML2 Web App

SAML Protocol Configuration Parameters

- **SAML Version:** 2.0
- **Issuer:** `urn:demo314.us.auth0.com`
- **Identity Provider Certificate:** Download Auth0 certificate
- **Identity Provider SHA1 fingerprint:** `B4:B7:5D:2F:70:44:D2:C6:AE:3B:27:33:78:B1:0F:1E:AB:11:B8:C1`
- **Identity Provider Login URL:** https://demo314.us.auth0.com/samlp/A1nNbtosqK5xGbBRH6JggvmiJHFHsQHp
- **Identity Provider Metadata:** Download

Alternatively, you can add a connection parameter:

- https://demo314.us.auth0.com/samlp/A1nNbtosqK5xGbBRH6JggvmiJHFHsQHp?connection=facebook
- https://demo314.us.auth0.com/samlp/A1nNbtosqK5xGbBRH6JggvmiJHFHsQHp?connection=google-oauth2
- https://demo314.us.auth0.com/samlp/A1nNbtosqK5xGbBRH6JggvmiJHFHsQHp?connection=Username-Password-Authentication

In this case, Auth0 will redirect users to the specified `connection` and will not display the Login Widget. Make sure you send the SAMLRequest using `HTTP POST`.

Finally, follow Asana's instructions to configure SSO on your Asana instance.

At the end of this setup, Asana and Google Workspace will each be using Auth0 as their Identity Provider. Now users in your Auth0 tenant can log in to both Google Workspace and Asana with one set of credentials.

Furthermore, if a user enters their credentials to access either application, they do not need to re-enter them when accessing the other — welcome to the world of Single Sign-On!

Auth0 as SAML IdP and Service Provider

Now it's time to bring your custom application to the SAML-based Single Sign-On party!

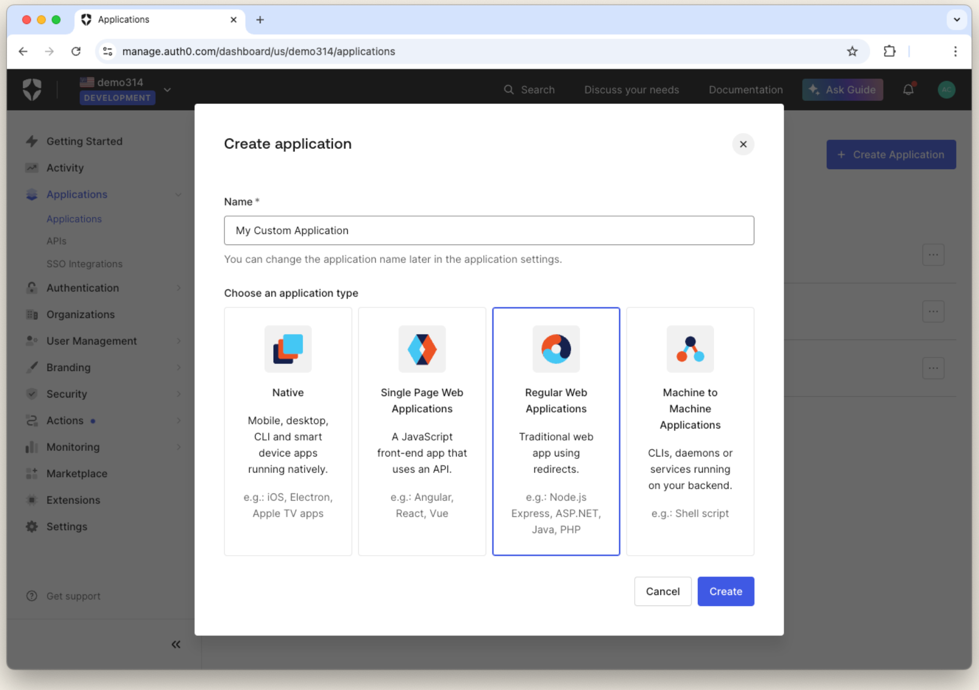
As we mentioned earlier, in this case Auth0 not only needs to act as an Identity Provider, but also as a Service Provider in order to connect the authentication result with your application. You will enable Auth0 to work in this dual capacity using two distinct tenants:

- One intended for the IdP function
- One for the Service Provider function.

Configure the IdP tenant

You will use the tenant where you have already configured Google Workspace and Asana as the IdP tenant for your custom application. Basically, you will run the same steps you used to configure your Asana instance in this tenant.

In your Auth0 dashboard, navigate to Applications > Applications and click **Create Application**. Enter a friendly name for your application (such as **My Custom Application**), and click the **Create** button:



Create application

Name *

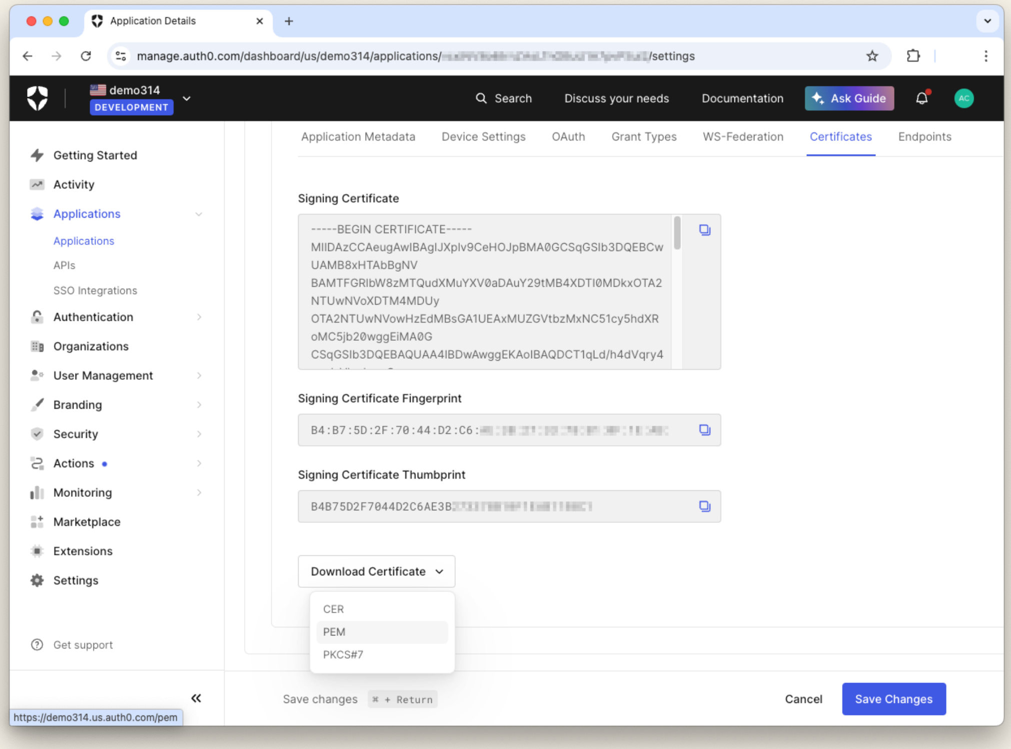
My Custom Application

You can change the application name later in the application settings.

Choose an application type

Native	Single Page Web Applications	[✓ Regular Web Applications]	Machine to Machine Applications
Mobile, desktop, CLI and smart device apps running natively.	A JavaScript front-end app that uses an API.	Traditional web app using redirects.	CLIs, daemons or services running on your backend.
e.g.: iOS, Electron, Apple TV apps	e.g.: Angular, React, Vue	e.g.: Node.js Express, ASP.NET, Java, PHP	e.g.: Shell script

Now go to the Settings tab and expand the **Advanced Settings** section down in the page. Then, switch to the **Certificates** view, click **Download Certificate**, and select **PEM** as the certificate format:



Signing Certificate

```
-----BEGIN CERTIFICATE-----
MIIDAzCCAeugAwIBAgIJXplv9CeHOJpBMA0GCSqGSIb3DQEBCw
UAMB8xHTAbBgNV
BAMTFGRlbW8zMTQudXMuYXV0aDAuY29tMB4XDTI0MDkxOTA2
NTUwNVoXDTM4MDUy
OTA2NTUwNVowHzEdMBsGA1UEAxMUZGVtbzMxNC51cy5hdXR
oMC5jb20wggEiMA0G
CSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDCT1qLd/h4dVqry4
```

Signing Certificate Fingerprint

`B4:B7:5D:2F:70:44:D2:C6:`[redacted]

Signing Certificate Thumbprint

`B4B75D2F7044D2C6AE3B`[redacted]

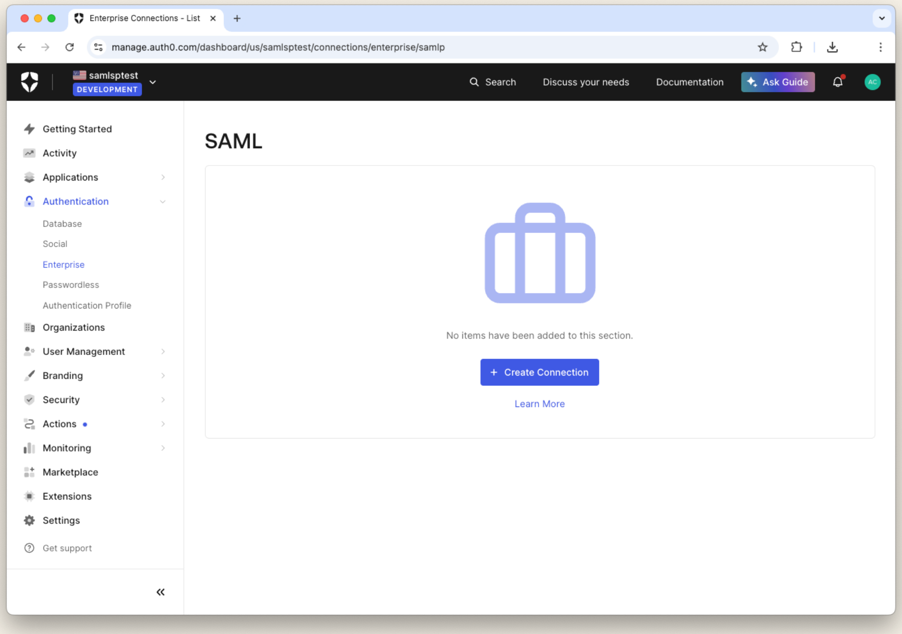
You will use the downloaded certificate when you configure the Service Provider tenant. Remember that the file containing the certificate will have the name of your current tenant.

Now go to the **Endpoints** tab, locate the **SAML** section and take note of the **SAML Protocol URL** shown there.

Configure the Service Provider tenant

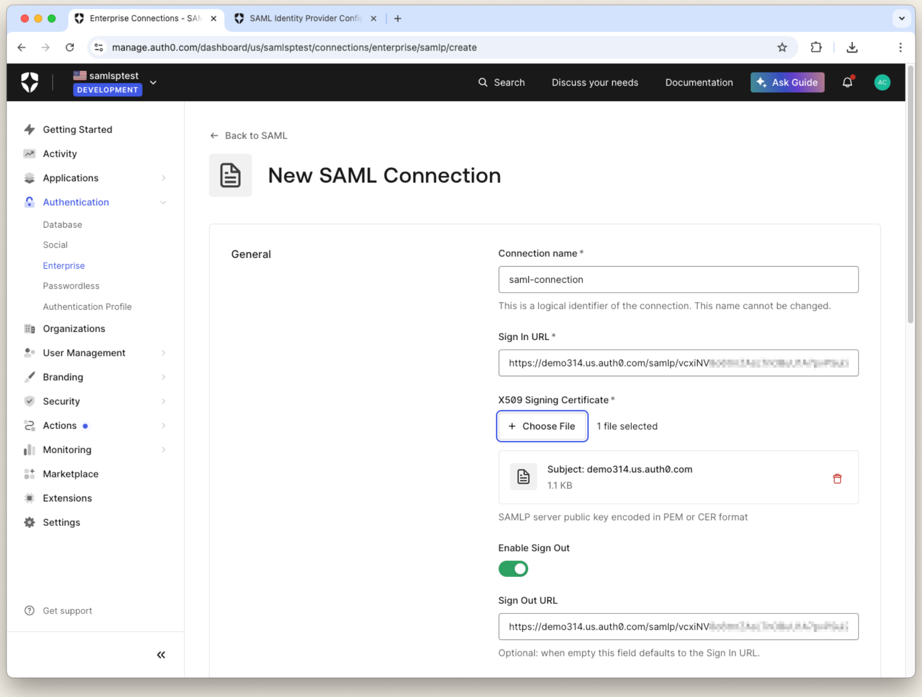
Switch to the tenant where you are going to configure Auth0 as a SAML Service Provider. You can also create a new tenant for this reason.

In your Auth0 Service Provider tenant, go to Authentication > Enterprise, select **SAML**, and then click the **Create Connection** button:



No items have been added to this section.

You will get a form where you'll enter the configuration data for your SAML connection:



New SAML Connection

General

Connection name *
saml-connection
This is a logical identifier of the connection. This name cannot be changed.

Sign In URL *
https://demo314.us.auth0.com/samlp/vcxiNV[redacted]

X509 Signing Certificate *
1 file selected
Subject: demo314.us.auth0.com
1.1 KB
SAMLP server public key encoded in PEM or CER format

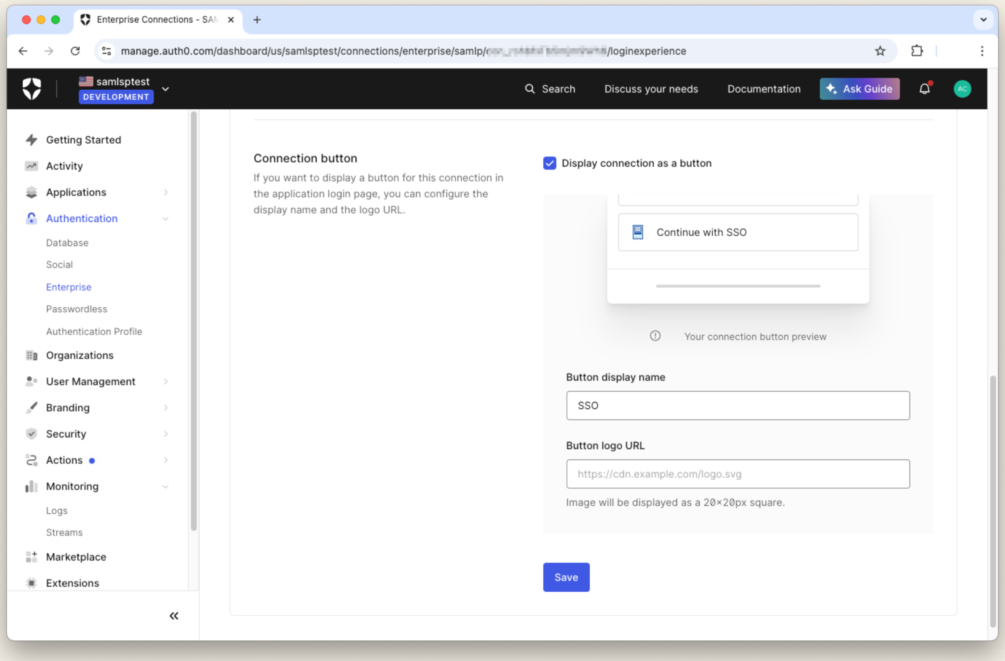
Enable Sign Out
[✓]

Sign Out URL
https://demo314.us.auth0.com/samlp/vcxiNV[redacted]
Optional: when empty this field defaults to the Sign In URL.

Assign a name to your SAML connection, such as **saml-connection** (and make sure you don't use spaces or other special characters for your SAML connection name).

Enter the **SAML Protocol URL** you copied earlier into both the **Sign in URL** and **Sign Out URL** fields. Upload the PEM file with the SAML certificate of your Auth0 IdP tenant. Finally, click the **Create** button.

In the resulting SAML configuration page, go to the **Login Experience** tab. Here you can choose how your company users will be requested to authenticate using SAML-based SSO. For our purposes, let's show them a button for logging in. Locate the **Connection button** section in the **Login Experience** tab and check **Display connection as a button**. Then assign the text you want to display on the button as the authentication method to the **Button display name** field. In our example, you will assign **SSO**, so that the login button will display "Continue with SSO". The following picture shows the **Login Experience** configuration:


manage.auth0.com/dashboard/us/samlsptest/connections/enterprise/samlp/[redacted]/loginexperience

Connection button

If you want to display a button for this connection in the application login page, you can configure the display name and the logo URL.

☑ Display connection as a button

Continue with SSO

Your connection button preview

Button display name: SSO

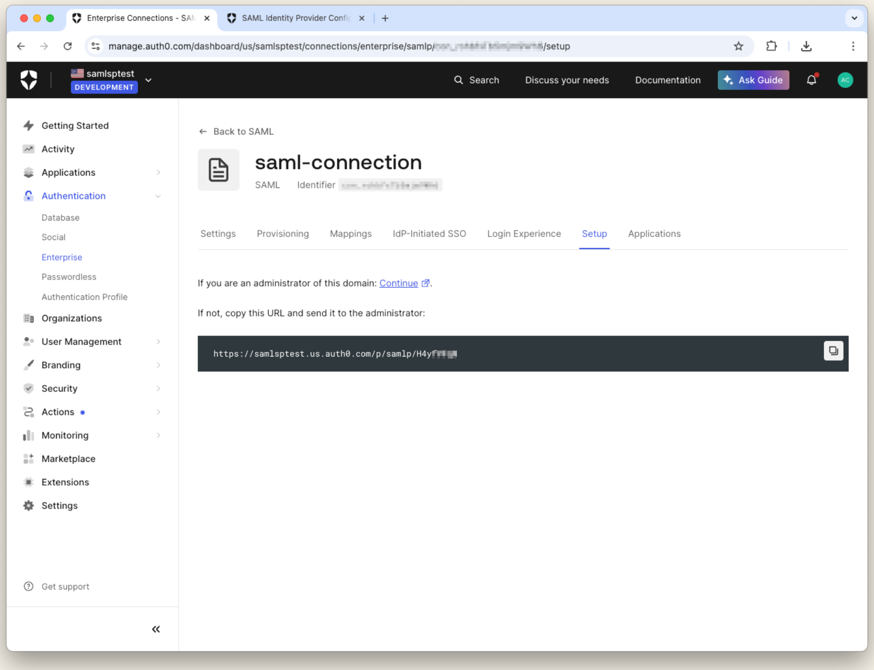
Button logo URL: https://cdn.example.com/logo.svg

Image will be displayed as a 20×20px square.

Save

Don't forget to click **Save** to confirm your settings.

Now, switch to the **Setup** tab of the **SAML configuration page**:



If you are an administrator of this domain: Continue.

If not, copy this URL and send it to the administrator:

```
https://samlsptest.us.auth0.com/p/samlp/H4y[redacted]
```

In this screen, you will see a URL that will bring you to a documentation page with all the details of the Service Provider you may need to provide your SAML-based Identity Provider. For our simple use case, you will need to take note of the following data:

- The **Entity ID**, which is a string in the URN format identifying your SAML connection. Its structure will be as follows:

  `urn:auth0:{YOUR-TENANT-NAME}:{YOUR-SAML-CONNECTION-NAME}`

  If your tenant name is `foo` and your SAML connection name is `saml-connection`, your Entity ID will be `urn:auth0:foo:saml-connection`.

- The **Assertion Consumer Service URL**, which is the URL on the Service Provider that receives the SAML assertion from the IdP tenant, as you already may know. It has the form `https://{YOUR-SP-DOMAIN}/login/callback?connection={YOUR-SAML-CONNECTION-NAME}`.

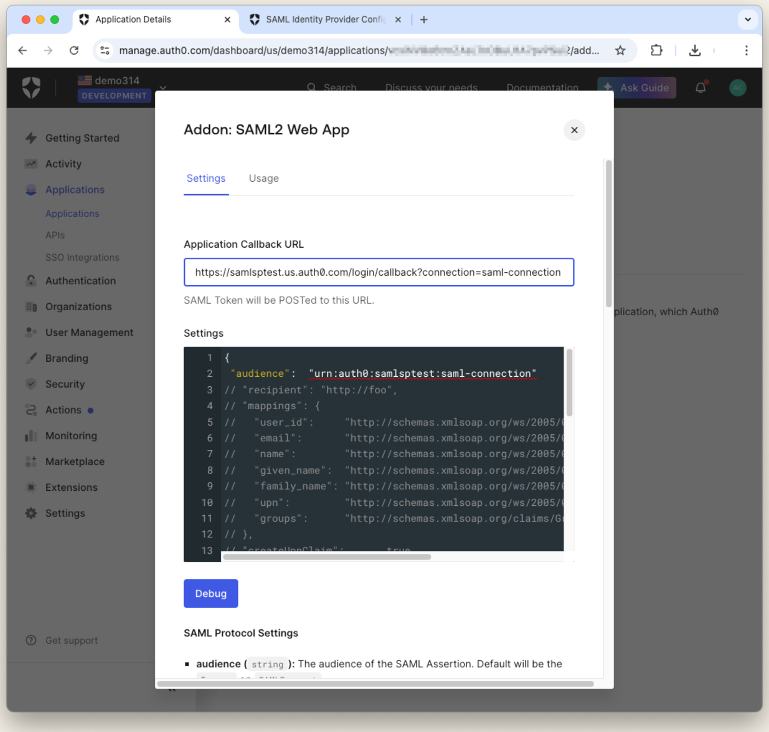
If the domain of the tenant for the Service Provider you are configuring is `foo.auth0.com` and your SAML connection name is `saml-connection` , your Assertion Consumer Service URL will be `https://foo.auth0.com/login/callback?connection=saml-connection`.

Now you have all the data to complete the configuration of your Auth0IdP tenant.

Complete the IdP configuration

Go back to your Auth0 IdP tenant, and navigate to Applications > Applications. Here, select the application you created for your custom application, **My Custom Application**, if you used the name suggested in the previous section.

Once you're in the **Application Configuration** page, select the **Addons** tab and click the **SAML2 Web App** item. You are about to do the same configuration steps you did when you configured Asana for SSO:



Addon: SAML2 Web App

Application Callback URL

https://samlsptest.us.auth0.com/login/callback?connection=saml-connection

SAML Token will be POSTed to this URL.

Settings

```
{
 "audience":  "urn:auth0:samlsptest:saml-connection"
// "recipient": "http://foo",
// "mappings": {
//   "user_id":     "http://schemas.xmlsoap.org/ws/2005/
//   "email":       "http://schemas.xmlsoap.org/ws/2005/
//   "name":        "http://schemas.xmlsoap.org/ws/2005/
//   "given_name":  "http://schemas.xmlsoap.org/ws/2005/
//   "family_name": "http://schemas.xmlsoap.org/ws/2005/
//   "upn":         "http://schemas.xmlsoap.org/ws/2005/
//   "groups":      "http://schemas.xmlsoap.org/claims/G
// },
```

SAML Protocol Settings

- **audience (** `string` **):** The audience of the SAML Assertion. Default will be the

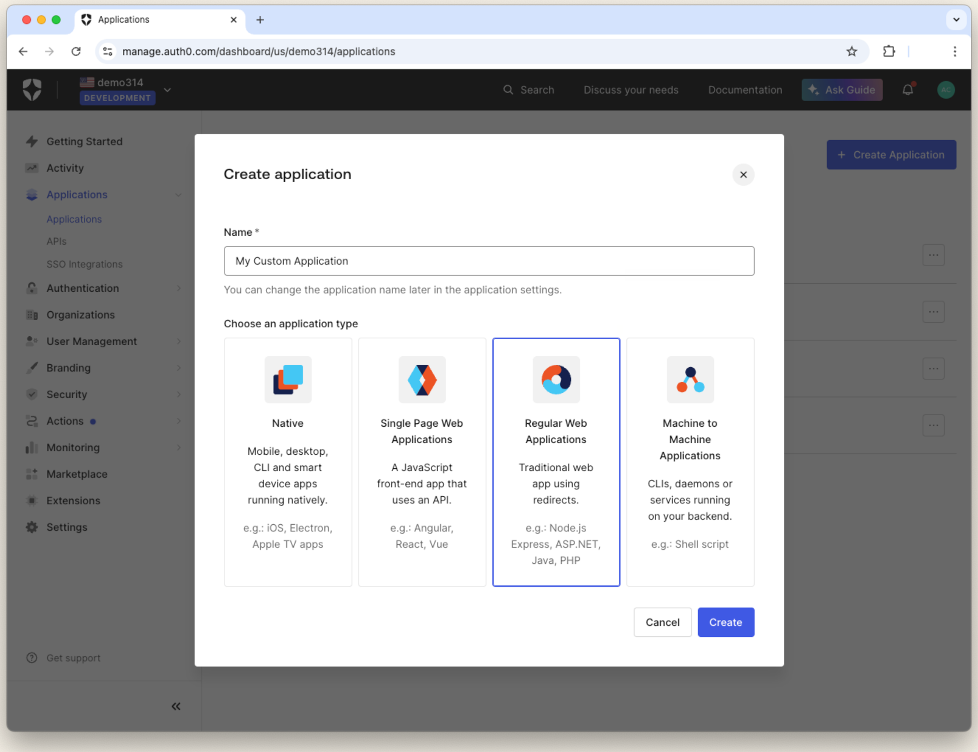
Enter the **Assertion Consumer Service URL** of your Service Provider tenant in the **Application Callback URL** field.

In the **Settings** code block, uncomment the audience key and replace its assigned value with the **Entity ID** of your Service Provider (don't forget to remove the trailing comma!).

Scroll down and click **Enable** to confirm your changes. Your custom application has been configured to use Auth0 as its SAML-based Identity Provider.

Register your custom application
At this point, Auth0 is configured as an Identity Provider and a Service Provider and is ready to carry out its tasks. You are just missing one final step: registering your custom application with Auth0 in the Service Provider tenant.

Go back to your Service Provider tenant, navigate to Applications > Applications, and click **Create Application**:



Create application

Name *

My Custom Application

You can change the application name later in the application settings.

Choose an application type

Native	Single Page Web Applications	Regular Web Applications	Machine to Machine Applications
Mobile, desktop, CLI and smart device apps running natively.	A JavaScript front-end app that uses an API.	Traditional web app using redirects.	CLIs, daemons or services running on your backend.
e.g.: iOS, Electron, Apple TV apps	e.g.: Angular, React, Vue	e.g.: Node.js Express, ASP.NET, Java, PHP	e.g.: Shell script

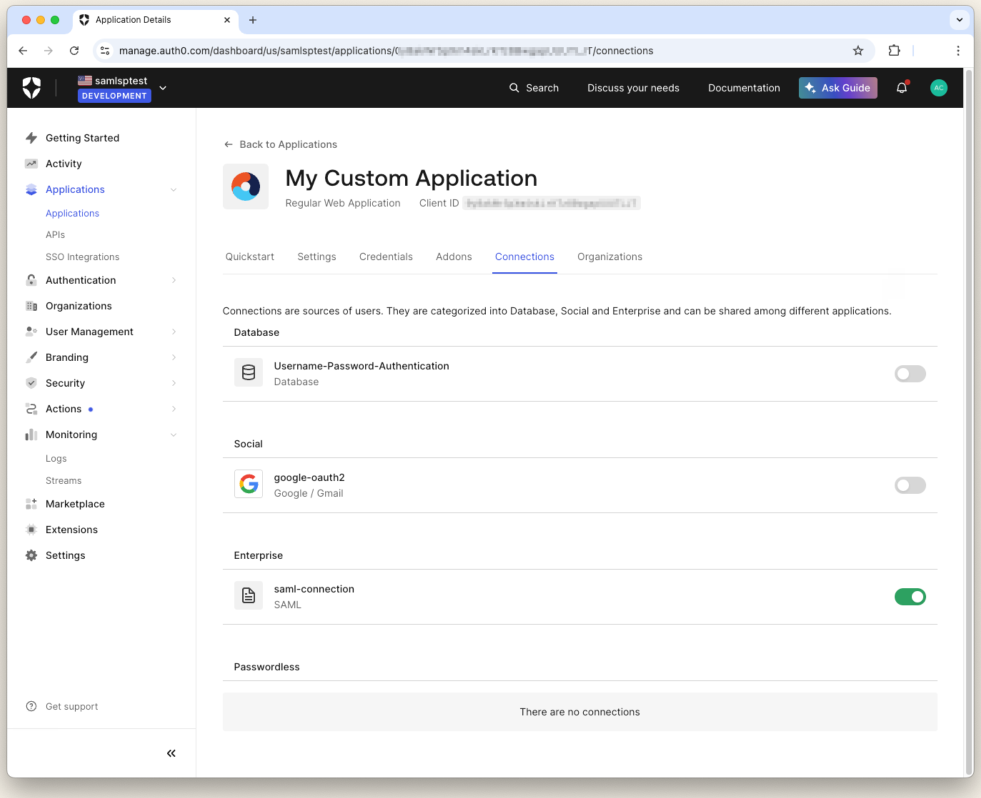
Enter a friendly name for your application and click the **Create** button.

In the application configuration page, go to the **Settings** tab and take note of the values of the **Domain** and **Client ID** fields.

Scroll down to the **Application URLs** section and enter the value for the **Allowed Callback URLs** field. This is the URL where your custom application will receive the token that confirms that the user was authenticated. Its value depends on how your application is built. Since we are going to use a sample application for this guide, set its value to `http://localhost:3000/callback`.

Click the **Save Changes** button to confirm your changes.

Then, switch to the **Connections** tab and enable the SAML connection you created earlier. Assuming you want to allow users to access your application using only SSO, you should also disable all the other connections, as shown below:



My Custom Application

Regular Web Application Client ID [redacted]

Connections are sources of users. They are categorized into Database, Social and Enterprise and can be shared among different applications.

Database

Username-Password-Authentication
Database

Social

google-oauth2
Google / Gmail

Enterprise

saml-connection
SAML

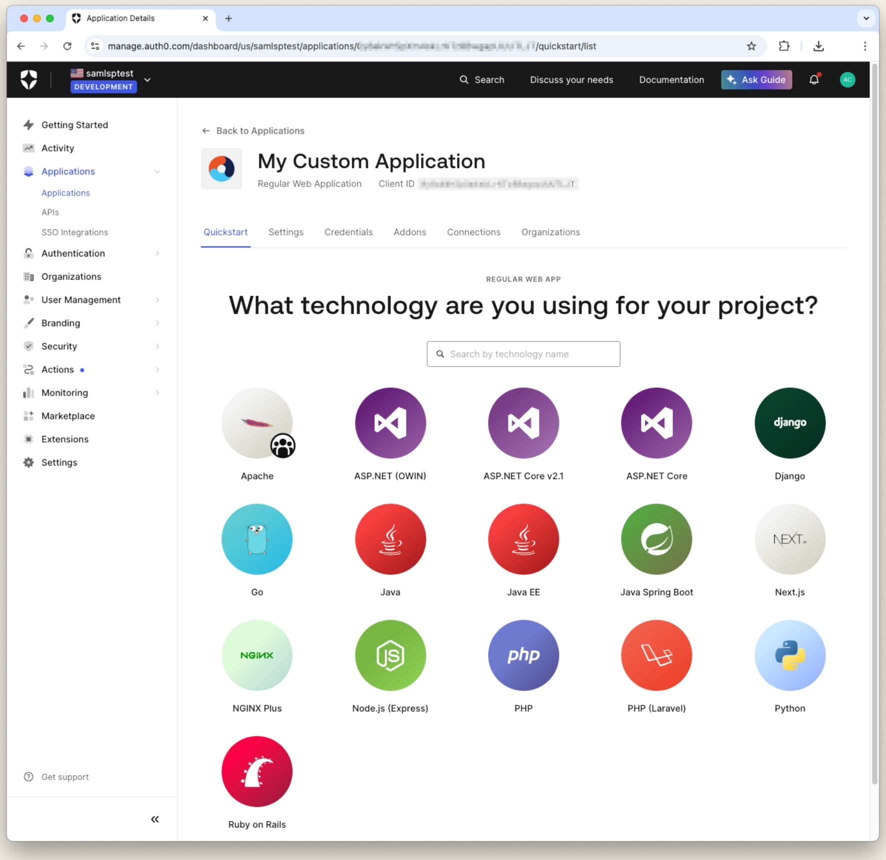
Passwordless

There are no connections

Test the SSO integration

Finally, you need to configure your custom application with the settings you got from the previous step. Basically, you need to provide your application with the domain and the client ID and enable it to call Auth0 for user authentication. How to implement this depends on the specific technology stack you used to build your application. Take a look at the Quickstarts to find the right instructions for your application.

For testing purposes, you can download a configured sample application from the **Quickstart** tab of the application configuration page.



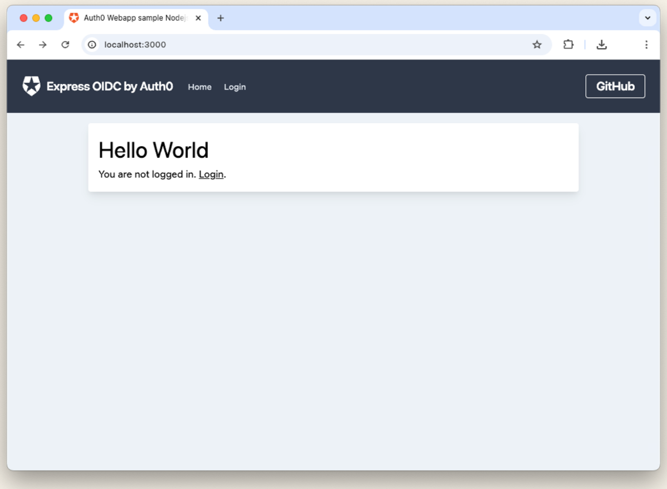
Choose your preferred technology and follow the instructions to download and get the application ready to run.

For this guide, we will download the **Node.js** web application. Once you get the sample application, go to its folder and run the following commands:

`npm install`

`npm start`

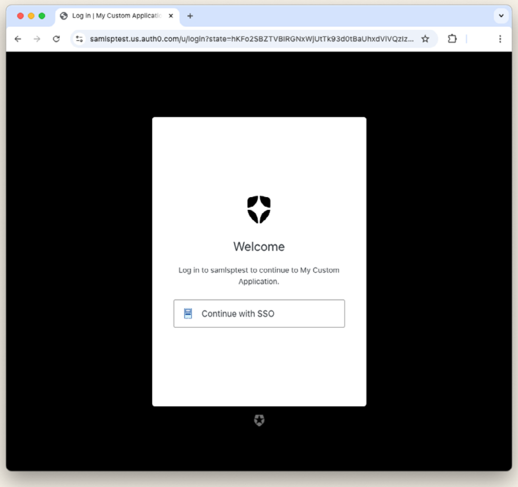
Then, point your browser to `http://localhost:3000/` and you should see the following page:



Hello World

You are not logged in. Login.

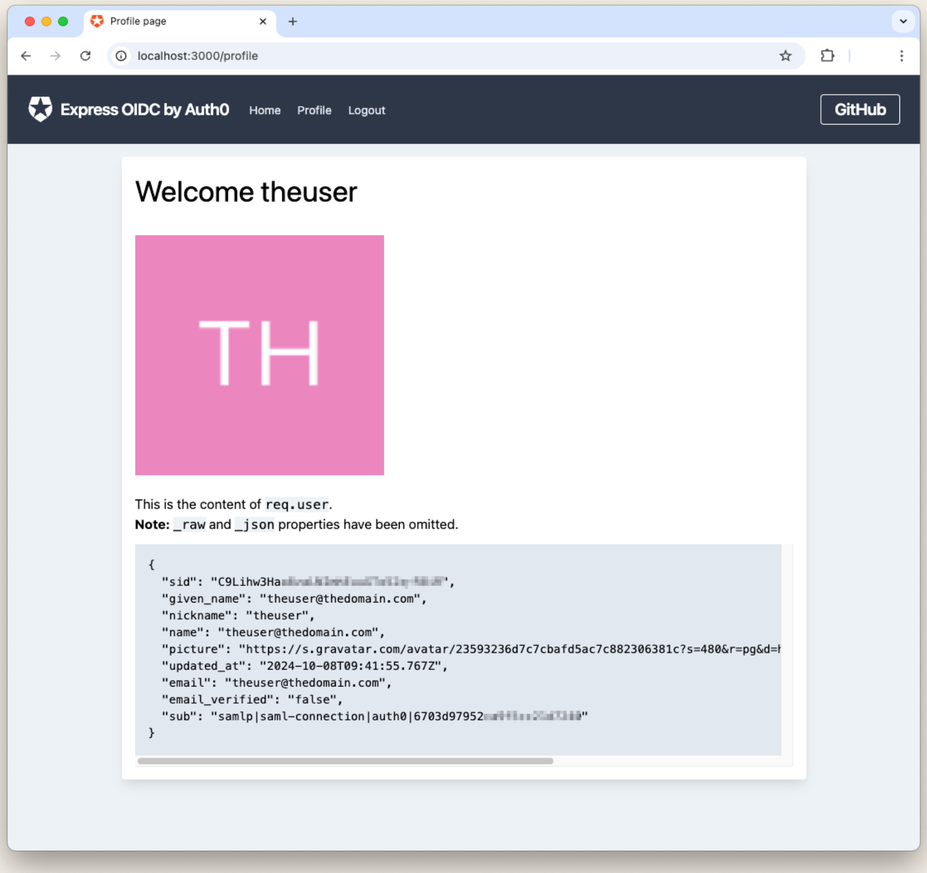
Once you click the Login link, you are redirected to the Service Provider tenant that shows you the login button you configured for the SAML connection:


Log in | My Custom Applicatio
samlsptest.us.auth0.com/u/login?state=hKFo2SBZTVBlRGNxWjUtTk93d0tBaUhxdVlVQzlz...
Continue with SSO

Welcome

Log in to samlsptest to continue to My Custom Application.

By clicking **Continue with SSO**, you will access the application if you have already been authenticated. Otherwise, you will need to provide your credentials to access the application. The **Profile** page in the application will display the details about the authenticated user:



Welcome theuser

This is the content of `req.user`.
Note: `_raw` and `_json` properties have been omitted.

```
{
  "sid": "C9Lihw3Ha[redacted]",
  "given_name": "theuser@thedomain.com",
  "nickname": "theuser",
  "name": "theuser@thedomain.com",
  "picture": "https://s.gravatar.com/avatar/23593236d7c7cbafd5ac7c882306381c?s=480&r=pg&d=h
  "updated_at": "2024-10-08T09:41:55.767Z",
  "email": "theuser@thedomain.com",
  "email_verified": "false",
  "sub": "samlp|saml-connection|auth0|6703d97952[redacted]"
}
```

Congratulations — you have successfully set up SAML-based Single Sign-On for the three applications currently used in your company!

Conclusion and next steps

Single Sign-On is simple to use, can be complex to implement, and is becoming more important with every new app that's developed.

The goals of this guide were to help you to:

- Understand what Single Sign-On is and how it works,
- Know the benefits of using Single Sign-On within your organization,
- Learn why potential customers value (or even require) SSO support,
- Become familiar with the various Identity protocols supporting SSO,
- Implement Single Sign-On in a guided tutorial.

Yet, for all that we've covered, we've still only scratched the surface of SSO.

If this guide has convinced you that SSO can benefit your organization — or has at least caused you to give it some thought — why not give it a try?

Ready to get started? We can help.

With 10+ years as a pioneer in the Customer Identity and Access Management (CIAM) space, Auth0 by Okta has leveraged cutting-edge technologies and architectures to create a modern solution with an impressive collection of features and functions, including:

- Performance tiers to support varying business needs, plus discounted (but still feature-rich) offerings
- Customizable actions, and pre-built partner integrations
- 60+ social and Identity Provider connections
- Broad support for open standards: OIDC, SAML, FIDO, OAuth, and more
- Multi-cloud encryption architecture and support for running on AWS and Azure — including public and private cloud options
- Certain features and functionalities that align with HIPAA or PCI requirements
- Easily customizable and brandable sign-up and login interfaces and flows

Choosing Auth0 by Okta means your CIAM needs can be met today, tomorrow, and as far into the future as you can imagine — because we do the hard work of:

- Keeping up with specifications and standards, which makes adding new interoperability and delivering the latest features — like passkeys — as simple as toggling a configuration variable
- Setting the stage for how we can support you with your new and evolving regulatory requirements
- Researching threat actor tactics, techniques, and procedures (TTPs) and introducing the advanced security features needed to combat these threats
- Exploring how new technologies — such as generative artificial intelligence (AI) — can be applied to Customer Identity

All of this means two big things for our customers:

1. Their Customer Identity capabilities are always state-of-the-art
2. They can apply their finite resources on their core value proposition

Speed up your Customer Identity journey with Auth0 SaaStart
Getting started with Customer Identity — especially for B2B SaaS apps — can be intimidating. Plus, developers are already under enough time constraints and don't always have the luxury of building sandboxes or proofs-of-concept from scratch.

Nevertheless, there's no substitute for learning through hands-on experience. That's why we created SaaStart — a reference B2B SaaS application built using Next.js and Auth0 by Okta.

Intended primarily for developers building B2B SaaS apps, SaaStart provides a fast, safe, and accessible way for devs to explore, experiment, and get up and running with Customer Identity.

- **Learn more about SaaStart**
- **Go to the SaaStart GitHub repository**

Explore our discounted — but still feature-rich — offerings
We're also proud to offer a number of discounted offerings that can help you get started and — thanks to a long list of features — are often sufficient for the longer term:

- Self-service options, including a free plan that lets you get off the ground and unlimited logins — while also including social login, DB connection for securely storing Identity information, branded login, extensibility and security capabilities (brute force and suspicious IP throttling), support for five regions (US, UK, EU, JP, AUS), and even passkeys
- Auth0 for startups, which brings the convenience and security of Auth0 to eligible startup customers so they can focus on their core value propositions rather than spending more time than is necessary on Identity
- Our Nonprofit Offering, which provides preferential pricing that makes the leading Identity service even more accessible to organizations on a mission to make the world a better place.

Glossary

For a more exhaustive list of authentication and Identity terms and definitions, please refer to Auth0's Identity Glossary.

Access Management	The set of functions that allow you to manage users' Identity and their access rights to resources and applications.
Authentication	The process of verifying a user's Identity by validating they are who they claim to be. Most commonly achieved through a username/password combination, but the same principle applies to other forms of authentication like passkeys, secret links, biometric identification, etc.
Authorization	Specifying which resources a user (with a given Identity) should be allowed to access.
Back-end server	A server where part of an application runs and is not directly accessible by the user — where Auth0 authenticates users, for instance.
Claim	A declaration about a subject that is supposed to be true and trusted depending on the Identity Provider. This declaration could be an attribute such as name, role, or permission.
Client	An application that obtains information from a server for local use.
Credentials	Information that proves a user's Identity or specific status, such as username and password, security token, etc.
Domain	A network where all resources and users are linked to a centralized database on which all authentication and authorization takes place.

Factor	In authentication, a vector through which Identity can be confirmed. There are three basic categories— knowledge factors (a password, PIN, security answer), ownership factors (security token, ID card), and inherence factors (fingerprint, DNA, retinal scan).
Federated Identity Management	A system of shared protocols that allows user identities to be managed across organizations.
Federation Provider	An Identity Provider that provides Single Sign-On, consistency in authorization practices, attributes exchange practices, and user management practices between Identity Providers (issuers) and relying parties (applications).
Identificaiton	The process by which a user's information is received, collected, and taken up for authentication.
Identity Provider (IdP)	A website, app, or service responsible for coordinating identities between users and clients. An IdP can provide a user with identifying information and serve that information to services when the user requests access.
Kerberos	A ticket-based protocol for authentication built on symmetric-key cryptography.
Multi-factor Authentication (MFA)	An authentication process that takes into account multiple factors. Commonly used in reference to two-factor authentication, which most commonly appears in the form of an SMS code sent to supplement a user's username/password login.

OAuth

An open standard for authorization. Development began in 2006 as employees from companies like Twitter and Google saw the need for a set of shared protocols dictating how web services should authorize other web apps to access their users' information. At its most simple, it works like this:

1. The user is prompted to authorize the client, or not, for a specific need (access to your Facebook friends list, say).
2. Proof of that authorization is sent to an (external) authentication server.
3. The authentication server gives the client a token representing access to the user's friends list.

OpenID Connect (OIDC)

An open standard for authentication. Allows third-party services to verify that users are who they say they are without clients needing to collect, store, and therefore become liable for a user's login information. At its most simple, it works like this:

1. The user selects an OIDC option upon login.
2. The client sends an authentication request to an external server (e.g., Facebook, Google, Twitter, etc.).
3. The external server verifies the Identity of the user, sending proof to the user if successful.
4. The user sends proof of authentication to the client.
5. Client approves or denies access.

Passwordless

A form of authentication that does not require the user to memorize a password. It's based on tokens, most commonly received and sent through SMS, email (magic links) or biometric sensors. Entirely based on inherence and ownership factors, passwordless is more secure than traditional username/password logins.

Role

An aspect of a user's Identity that gives them certain permissions.

Role-based Access Control (RBAC)	A model for authorization based on users gaining certain permissions based on their roles.
Security Assertion Markup Language (SAML)	An authentication and authorization standard commonly found in the enterprise. SAML differs from OpenID Connect in that it does not dynamically discover and accept authentication from new Identity Providers. The IdPs that a service wants to trust must be specified and hard-coded into each login event. Typically used to give the users of a corporate network access to a specific third-party service — for instance, so you don't have to sign in again when you click a link to Salesforce on your company's intranet.
Single Sign-On (SSO)	A subset of Federated Identity management, a means through which users can be authenticated once and access multiple applications.
Social Identity Provider (Social IdP)	A term used to refer to Identity Providers originating in social services like Facebook, Google, LinkedIn, etc.
Software as a Service (SaaS)	A model for software delivery and licensing in which software is accessed online via a subscription rather than bought and installed on individual computers.
WS-Federation	A federated standard or common infrastructure for Identity, used both by web services and browsers on Windows Identity Foundation.

Disclaimer

Okta Inc.
100 First Street
San Francisco, CA 94105
info@okta.com
1-888-722-7871