Auth0 Identity and Access Management Lab

Technical portfolio case study

Blake Cox | October 2026

Project overview

I replicated an Auth0 identity and access management tutorial from tenant setup through the final exercise. I configured SAML 2.0 single sign on, OpenID Connect authentication, JWT inspection, and OAuth 2.0 API authorization. All three protocol tests completed successfully, and I followed the tutorial without modifications.

This project demonstrates hands on application registration, federation configuration, and interpretation of authentication and authorization results. Auth0 served as the identity provider for SAML and as the OpenID Provider and authorization server for the token based exercises.

Exercise	Purpose	Protocol artifact
SAML 2.0	Federated sign in to a service provider	XML assertion
OpenID Connect	Authenticate a user to a client application	ID token
OAuth 2.0	Authorize access to an API	Access token

Tenant and application setup

I began by setting up the Auth0 development tenant and registering the applications used in the tutorial. I configured the SAML integration and the APP-OIDC client, then used the test interfaces to exercise the flows. APP-OIDC is registered as a Single Page Application, as shown below.

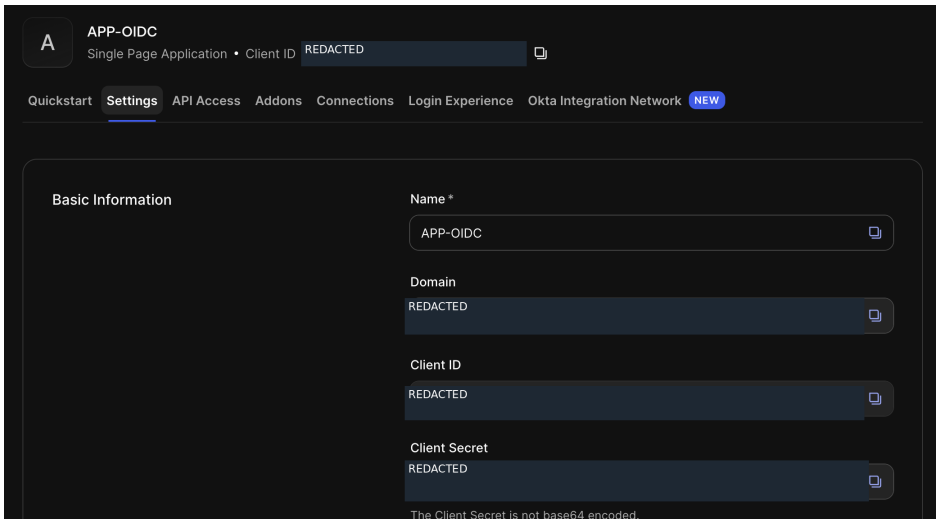


Figure 1 APP-OIDC registration in my Auth0 tenant. Tenant and client identifiers and the secret field have been removed for publication.

Project scope Tutorial based development lab. The implementation covers protocol demonstrations and test evidence rather than a production IAM deployment. The source tutorial is listed on page 4.

SAML 2 0 single sign on

I integrated Auth0 with the SAML Testing Tool at samlsp.com. Auth0 acted as the identity provider, authenticating the test user and returning a SAML response to the service provider. The service provider consumed the response and displayed the resulting identity attributes.

Configure the assertion destination

I used the service provider Assertion Consumer Service URL, https://samlsp.com/?acs, as the Application Callback URL in the Auth0 SAML2 Web App add on. The two screenshots below show the same endpoint on both sides of the integration. This alignment directs the response to the intended service provider. [2]

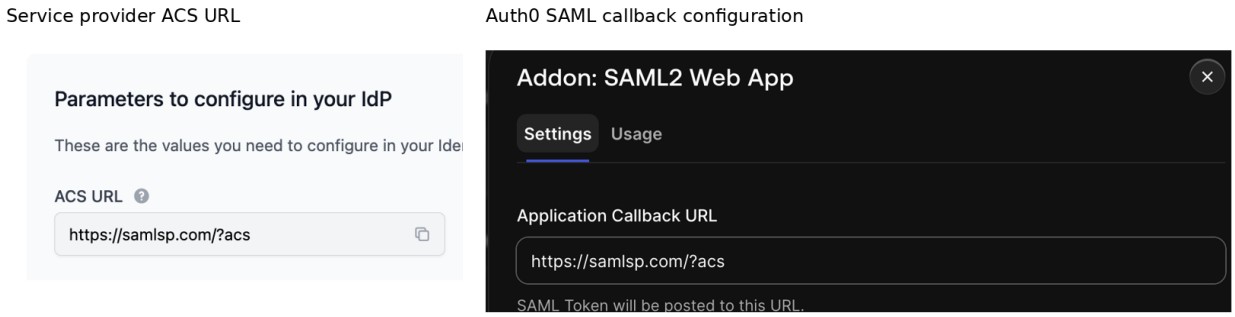


Figure 2 Matching service provider ACS and Auth0 callback settings.

Run the federated login

I initiated the SAML sign in and authenticated through Auth0. The resulting service provider page displays User Information, a NameID format, returned attribute names. The populated result page supports completion of the federated login.

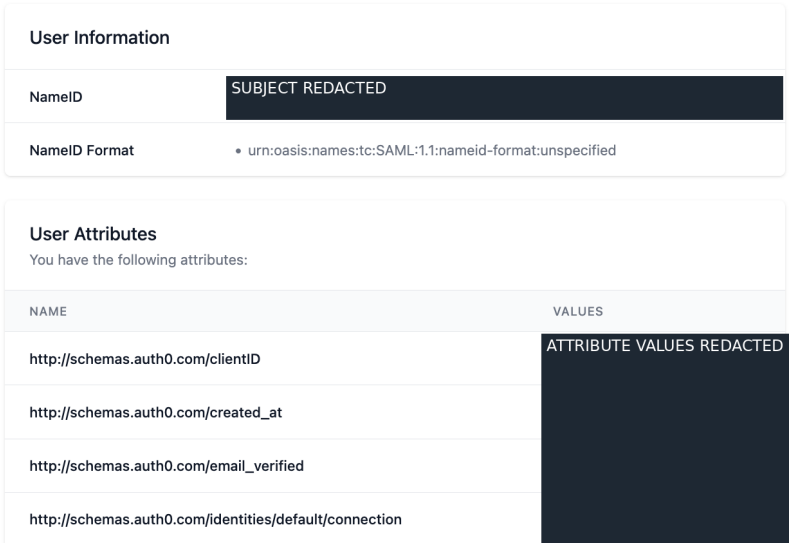


Figure 3 Detail of the SAML result after sign in. Subject and attribute values are masked; attribute names remain visible.

Evidence interpretation The result supports successful federation and delivery of identity attributes to the service provider. It does not separately document assertion signature verification, audience validation, or expired assertion rejection. Those checks remain requirements for a production integration.

OpenID Connect and JWT inspection

I configured the OIDC application and completed the tutorial authentication exercise using the OpenID Connect Debugger. OIDC adds an identity layer to OAuth 2.0 so that a client can receive an ID token describing an authenticated user. The ID token is intended for the client application. [3]

Interpret the authorization response

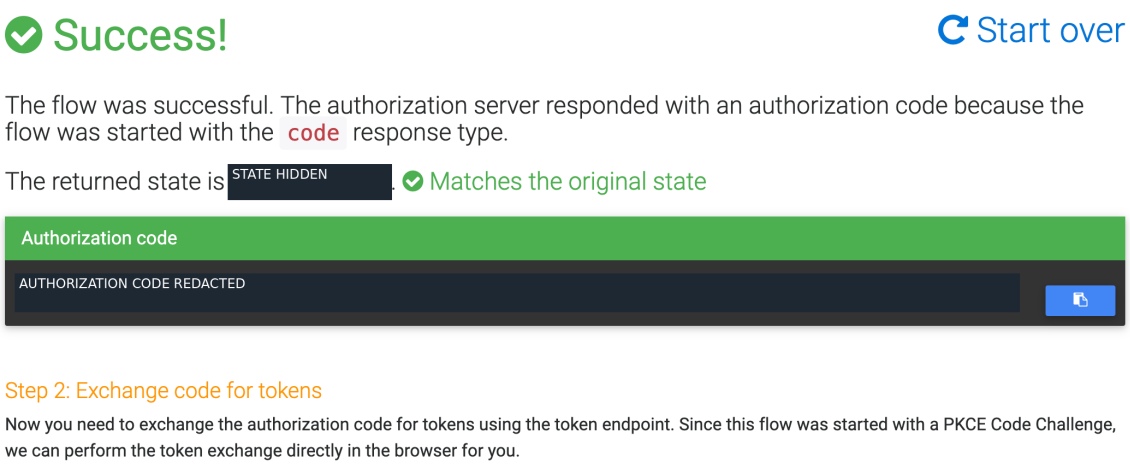


Figure 4 OIDC Debugger success response. The authorization code and state value are masked while the success and state matching indicators remain visible.

The debugger reports Success, identifies the response type as code, and confirms that the returned state matches the original state. It also states that the flow started with a PKCE Code Challenge. These details support successful completion of the authorization redirect and response correlation.

The next step displayed is Exchange code for tokens. This screenshot establishes the authorization response stage, before token exchange. I completed the full OIDC exercise successfully; the image does not show a decoded JWT.

Inspect the JWT structure

I inspected the header, payload, and signature of a signed JWT during the tutorial. These fields connect token content to trust decisions without publishing the original values.

Field	Meaning and security relevance
alg and kid	Identify the signing algorithm and, when present, the signing key. Accept only configured algorithms and trusted keys.
iss and aud	Identify the issuer and intended audience. Compare them with the values expected by the token recipient.
sub	Identifies the subject within the issuer context. Avoid exposing the original identifier in portfolio evidence.
iat and exp	Describe issuance and expiration times. Expiration must be enforced when accepting the token.

Decoding reveals content; validation requires checking the signature and applicable claims using a trusted library. Separate tampering and expiration tests were not documented. [4]

OAuth 2 0 API authorization

I completed the tutorial API authorization exercise successfully without modifying the workflow. The exercise connected an OAuth client, Auth0 as the authorization server, and the target API as the resource server. Its purpose was to demonstrate access authorization separately from user authentication.

An access token is the credential presented to an API. The API evaluates whether the token is acceptable for the requested resource and operation. An ID token serves the client authentication context and should not be substituted for an API access token. [3, 5]

The OAuth success is recorded from my completed test. The selected screenshots document SAML and OIDC; no API request, HTTP response, token payload, exact scope set, or OAuth grant type is reproduced here. The OIDC authorization code response is not used as evidence of a separate successful API call.

Testing results and evidence

Test	Result	Evidence and boundary
SAML 2.0 SSO	Passed	Matching ACS configuration and the service provider identity result in Figures 2 and 3.
OIDC authentication	Passed	Figure 4 shows code response success and matching state. Full exercise completion confirmed by me.
JWT inspection	Completed	Completed within the tutorial. Original token and decoded claim values are not reproduced.
OAuth 2.0 API authorization	Passed	Completion confirmed by me. No API response screenshot is included.

Practical lessons

The lab helped me connect application settings with protocol behavior. Matching the SAML callback to the ACS endpoint established the response destination. The OIDC debugger made the redirect, authorization code, state, and token exchange stages visible. JWT inspection clarified how identity and token metadata are represented, while the OAuth exercise reinforced the distinction between signing in and accessing an API.

For IAM work, these skills support application onboarding and troubleshooting of callback, audience, and token issues. Further validation could include invalid audience, expired token, and insufficient scope tests; those are future extensions rather than results claimed for this project.

Evidence handling

Published screenshots are cropped and flattened with opaque redactions. Authorization codes, state values, subject identifiers, tenant and client identifiers, and personal attribute values have been removed. No original tokens, passwords, or client secrets are embedded in this document.

References

[1] Source tutorial replicated from Auth0 tenant setup through the end. YouTube. https://youtu.be/jGz_e0sj_s4

[2] Auth0. Configure Auth0 as a SAML Identity Provider. https://auth0.com/docs/authenticate/single-sign-on/outbound-single-sign-on/configure-auth0-saml-identity-provider

[3] Auth0. ID Tokens. https://auth0.com/docs/secure/tokens/id-tokens

[4] Auth0. Validate JSON Web Tokens. https://auth0.com/docs/secure/tokens/json-web-tokens/validate-json-web-tokens

[5] Auth0. Access Tokens. https://auth0.com/docs/secure/tokens/access-tokens