**Introduction to Auth0 (Future Identity Platform for Berkley)**

**Auth0**

Auth0 is an identity management platform that provides authentication and authorization services for web, mobile, and other applications. It allows developers to easily integrate authentication features such as login, signup, and password management into their applications, without having to build these features from scratch. Auth0 supports various authentication methods including username/password, social logins (e.g., Google, Facebook, Twitter), and multi-factor authentication. Overall, Auth0 helps developers secure their applications and manage user identities efficiently.

**Note: WR Berkley will not support social logins.**

**Applications Supported by Auth0**

* **Regular Web Application**
* **Single Page Web Application**
* **Machine to Machine Application**

**Regular Web Applications**

Traditional web apps that perform most of their application logic on the server (such as Express.js or ASP.NET).

**Single Page Application**

JavaScript apps that perform most of their user interface logic in a web browser, communicating with a web server primarily using APIs (such as AngularJS + Node.js or React).

**Machine to Machine Applications**

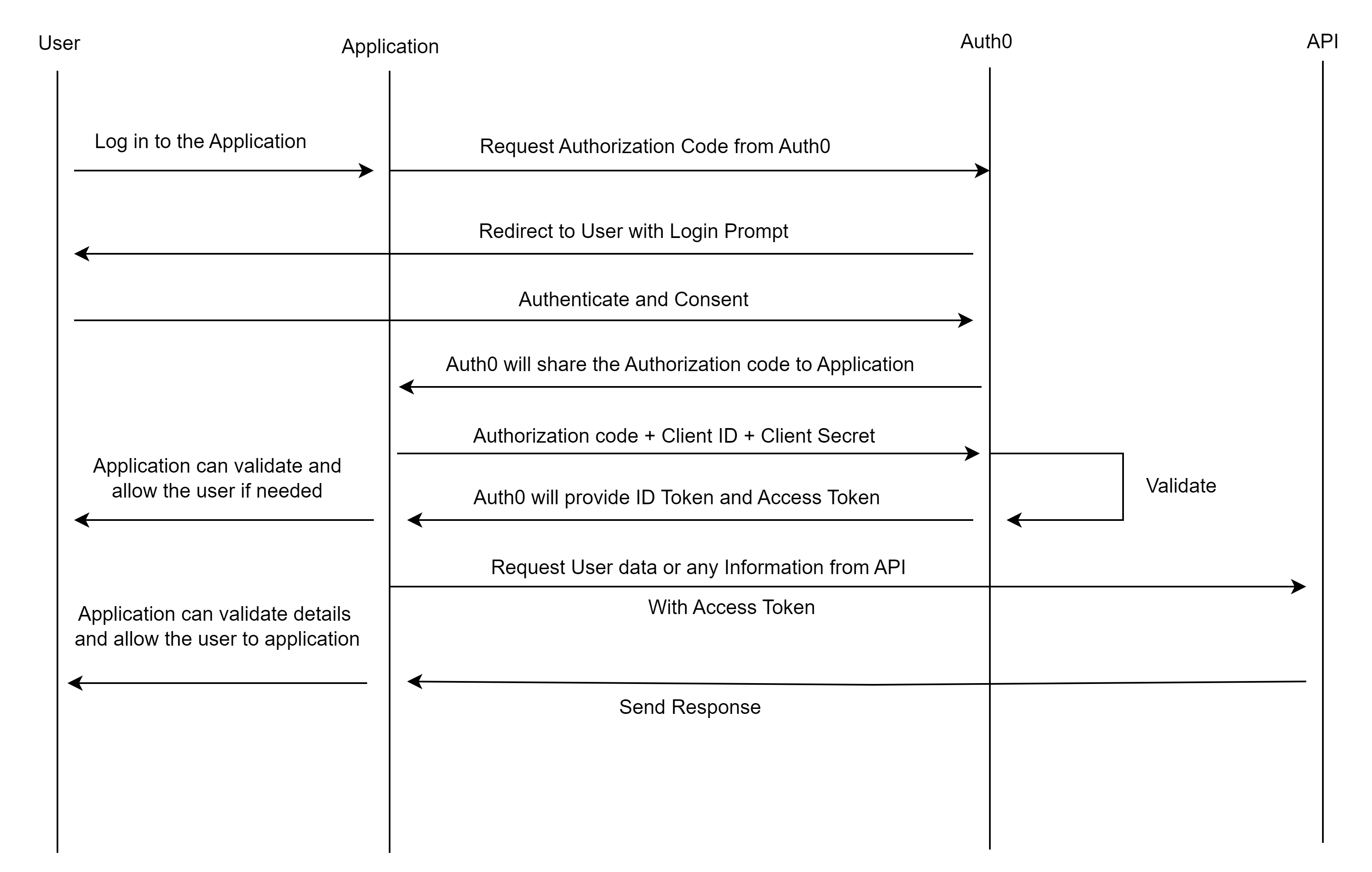
Non-interactive applications, such as command-line tools, IoT devices, or services running on your backend. Typically, you use this option if you have a service that requires access to an API.

**OAuth 2.0**

OAuth 2.0 is an authorization framework that allows third-party services to access a user's resources without exposing their credentials. The OAuth 2.0 flow refers to the sequence of steps that occur during the authorization process between a user, a client application, and a server that holds the user's resources.

**Authorization code flow**

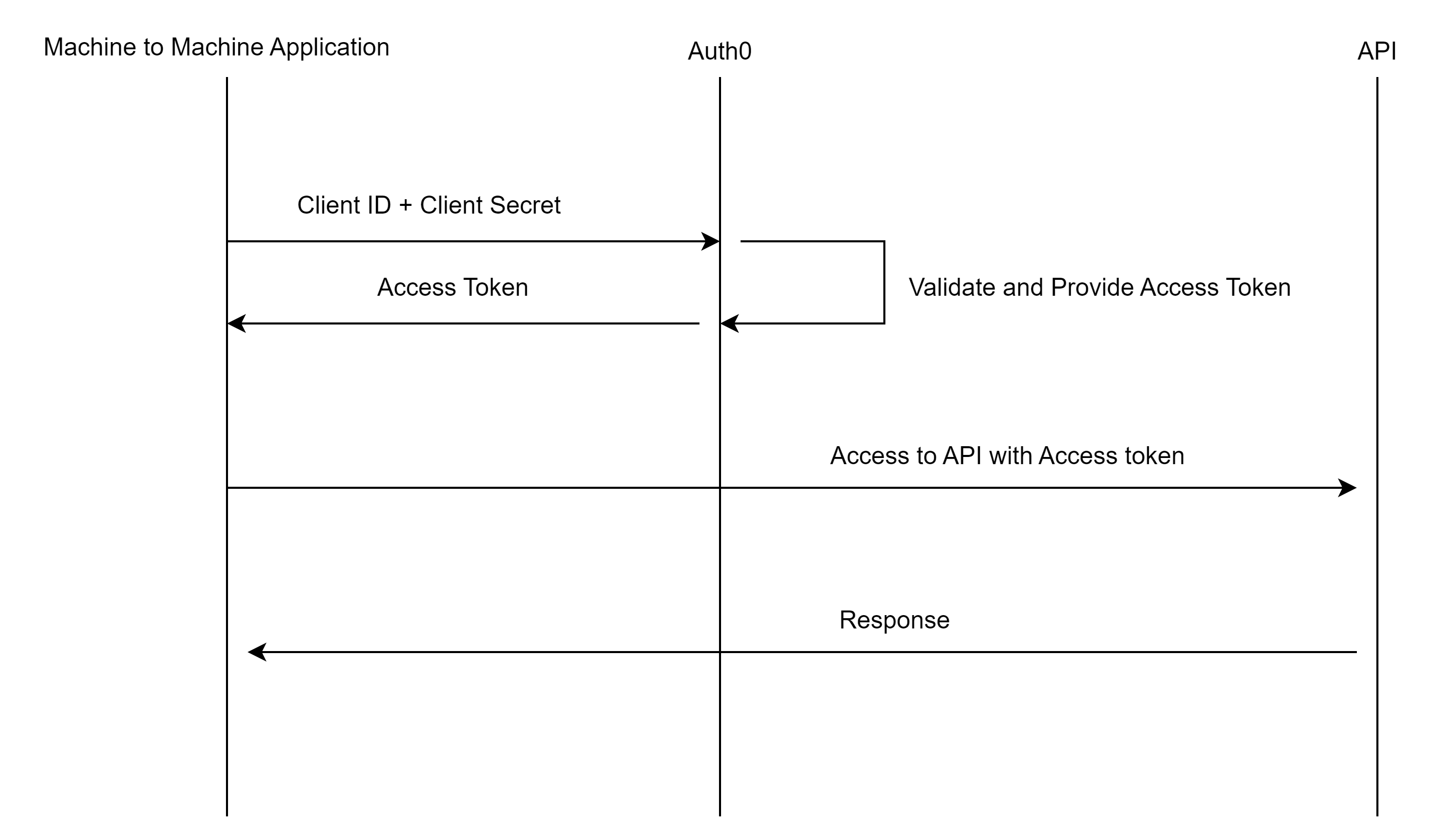
This flow is typically used for server-side applications where the client application can securely store a client secret. It involves multiple steps, including redirection to the authorization server, user authentication, authorization grant, and exchange of an authorization code for an access token. This flow is recommended for Regular Web applications.



**Note: KONG API Gateway is not Involved in the above picture. If the Application is going to use KONG, the flow picture will be different.**

**Client Credentials Flow:**

This flow is used when the client application is acting on its own behalf (not on behalf of a user) to access protected resources. It typically involves the client application directly exchanging its credentials (client ID and client secret) for an access token. This flow is recommended for Machine-to-machine Applications.



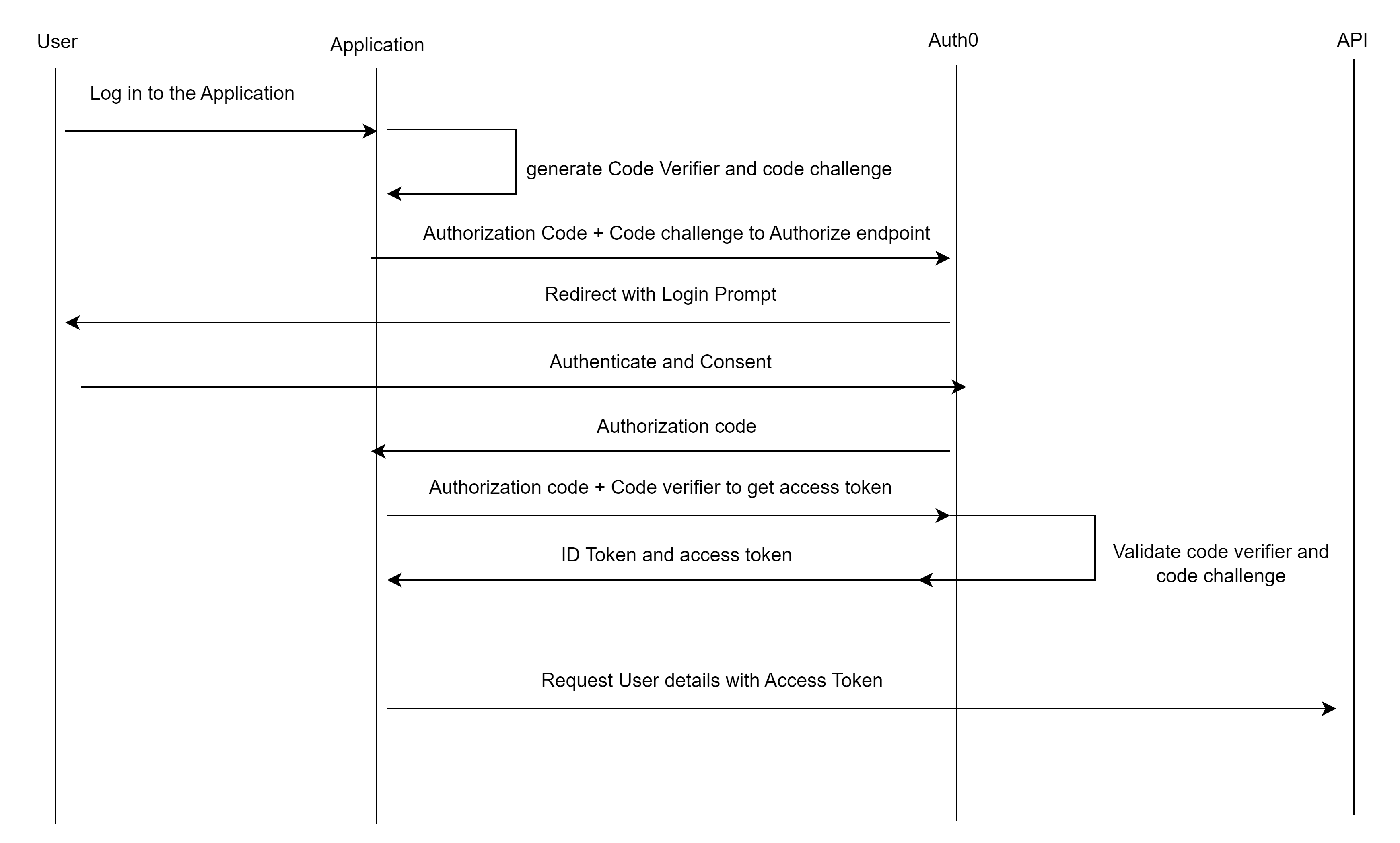
**Note: KONG API Gateway is not Involved in the above picture. If the Application is going to use KONG, the flow picture will be different.**

**Authorization code flow with PKCE**

This flow is designed to enhance security, particularly for mobile and native applications, where it's challenging to securely store a client secret. Mostly this flow is used for SPA (Single page applications).

**PKCE**

The Proof Key of Code Exchange (PKCE) is an extension of the standard authorization code grant OAuth flow. It is designed to be a secure substitute for the implicit flow for single-page applications (SPA) or native applications. SPA and native applications are vulnerable to reverse engineering practices.



**Note: KONG API Gateway is not Involved in the above picture. If the Application is going to use KONG, the flow picture will be different.**

**Authorization code**

An authorization code is a temporary code that a server issues to a client application as part of the process of granting authorization.

**Code Challenge and code verifier**

**Code Verifier Generation:** The client application generates a random string called the "code verifier". This string is typically a high-entropy cryptographic nonce (number used once), ideally generated using a secure random number generator.

**Code Challenge Generation:** The client application then creates a code challenge based on the code verifier. The code challenge is a transformed version of the code verifier, typically created using a cryptographic hash function (such as SHA-256). This transformation ensures that even if the code verifier is intercepted during transmission, the attacker cannot easily derive the original value.

**What is an ID Token?**

* An ID Token is a type of security token used primarily for identity confirmation.
* It is typically formatted as a JSON Web Token (JWT), which means it follows a specific structure and contains authenticated user profile information.
* When a user logs in via an authentication process like OIDC, they receive an ID token alongside an access token.

**Structure of an ID Token:**

**An ID token consists of several fields, each serving a specific purpose**

* **iss (Issuer):** Indicates the issuer of the token (e.g., the OAuth provider like Auth0). This field helps identify the authentication server that issued the token.
* **sub (Subject):** Represents a unique identifier for the user. It’s used to identify the entity (user) that the token pertains to.
* **aud (Audience):** Specifies the intended audience of the token (e.g., the application). The receiving service verifies that the token is intended for it.
* **exp (Expiration Time):** Indicates when the token expires. After this time, the token should not be accepted for processing.
* **iat (Issued At):** Indicates the timestamp when the token was created.

**What is an Access Token?**

* An Access Token is a crucial component of token-based authentication. It serves as a credential that allows an application to access protected resources (such as APIs) on behalf of an authenticated user.
* When a user successfully authenticates and authorizes access, the application receives an Access Token from Auth0.
* These tokens conform to the JWT standard and are self-contained. This means that the token itself contains all necessary information, and there’s no need to call a server for validation.
* JWT Access Tokens are used for scenarios like API authorization.
* They include standard claims (such as iss, sub, aud, exp, and iat) and custom claims specific to the application.

**Scopes in Access Token**

* The permissions represented by an Access Token are known as scopes.
* Scopes define what actions the application can perform at the API.
* For example, an Access Token may grant read or write access to specific resources.