Brief Introduction to OAuth

- Document Purpose
- OAuth Overview
- Overall Workflow of OAuth
  - Roles in the OAuth Workflow
  - High-Level Interaction Model
  - End-to-End Authorization Flow
- Typical Use Cases
  - Use Case 1: Granting Limited Access to Photos in Cloud Drive
  - Use Case 2: Sign in to a Third-Party Service with Existing Account
- Scope and Limitations
  - Scope
  - Limitations

Document Purpose

This example-driven document provides a high-level overview of how OAuth is used within a system to support secure authentication and authorization.

It is intended to help readers build a conceptual understanding of OAuth workflows, rather than serve as an implementation or integration guide.

Targe audience includes:

- Technical Writers

- Technical Support and Operation Engineers
- Software Developers

This document intentionally presents a conceptual view of OAuth workflows to support system-level understanding. Certain protocol details and implementation-specific variations are simplified or omitted.

OAuth Overview

OAuth is a delegation protocol that allows a resource owner to grant a client application limited access to protected resources on their behalf, without sharing account credentials or impersonating the user.
Throughout this document, a cloud photo printing scenario is used as a running example, where a user needs to authorize a third-party printing service to access selected photos stored in the cloud drive.

Overall Workflow of OAuth

Roles in the OAuth Workflow

During the authentication and authorization process of OAuth, the following roles are included:

Role	Description	Example
Resource Owner	The entity that has the authority to delegate access to the client. Usually a human being.	The end user
Client Application	A piece of software that attempts to access the protected resource on behalf of the resource owner, and it uses OAuth to obtain that access.	Photo printing service
Authorization Server	The server that authenticates the resource owner and issues access tokens after obtaining consent.	Cloud drive authorization service
Resource Server	The server that hosts protected resources and accepts access tokens to grant or deny access.	Cloud drive photo storage

High-Level Interaction Model

Fig.1 illustrates the high-level interaction model of the OAuth authorization code flow, using a photo printing service as an example.

The detailed workflow can be found in the next chapter.

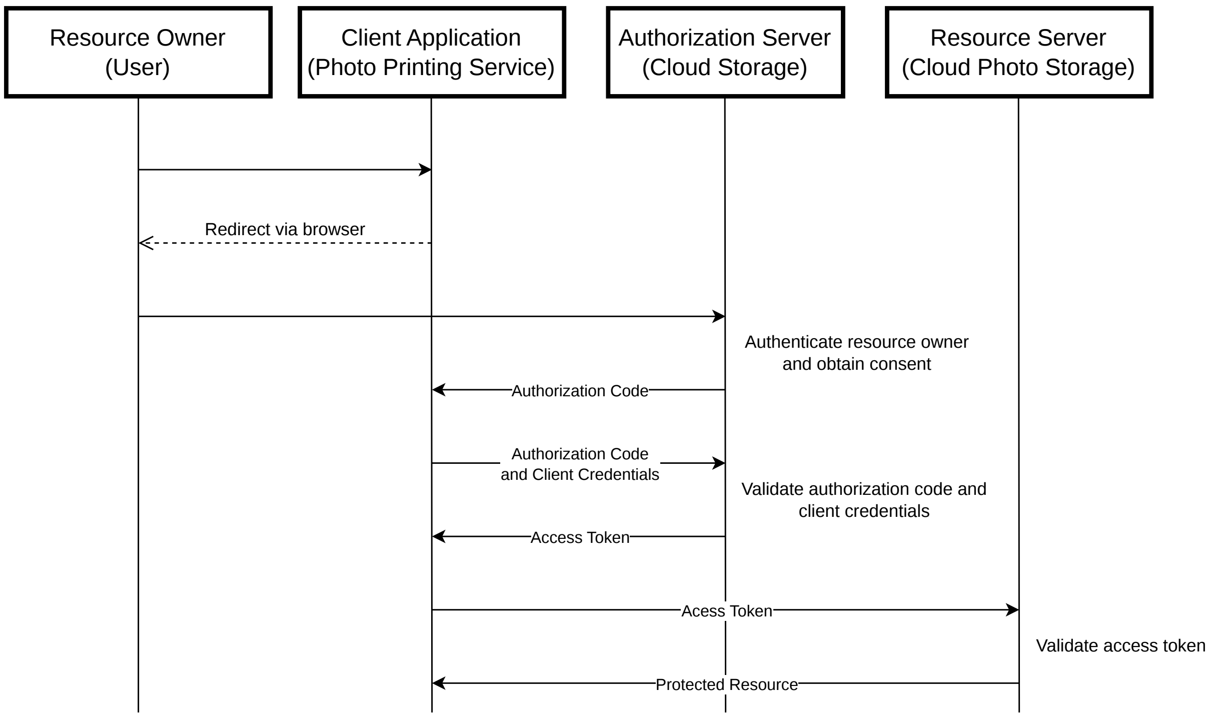
Resource Owner (User)
Client Application (Photo Printing Service)
Authorization Server (Cloud Storage)
Resource Server (Cloud Photo Storage)
Redirect via browser
Authenticate resource owner and obtain consent
Authorization Code
Authorization Code and Client Credentials
Validate authorization code and client credentials
Access Token
Acess Token
Validate access token
Protected Resource

Fig. 1 OAuth Authorization Code Flow

End-to-End Authorization Flow

1. Resource owner accesses the client application.
2. The client application redirects the resource owner's browser to the authorization server to request authorization.
3. The resource owner enters the credentials, and reviews the requested scopes.
4. The authorization server authenticates the resource owner and obtains the resource owner's consent.
5. The authorization server redirects the user's browser back to the client application with an authorization code.
6. The client application sends the authorization code, together with its client credentials, to the authorization server.

7. The authorization server validates the authorization code and client credentials, and issues an access token to the client application.
8. The client application sends a request to the resource server using the access token.
9. The resource server validates the access token and returns the requested protected resources to the client application.

Typical Use Cases

This section provides the common use cases where OAuth is used to provide secure, limited access to protected resources.

Use Case 1: Granting Limited Access to Photos in Cloud Drive

A user stores some photos in a cloud drive service, and finds a third-party photo printing service to print selected photos.

The original way for the printing service to retrieve the photos is using the user’s user name and password to get access to the photos, but all the other data can also be retrieved by the service, which may cause risks. On the contrary, with OAuth authorization flow, the printing service is only approved to get access to the approved photos.

This approach ensures that:

- The user’s account credentials are never shared with the third-party service.
- Access is limited to only the approved resources.
- Authorization is scoped to a specific service and purpose.

Use Case 2: Sign in to a Third-Party Service with Existing Account

A user already has an account with Company A, which manages the user’s identity and profile information. When the user wants to use a service provided by Company B, conventionally the user has to register a separate account.

But through OAuth-based authorization, Company B can request permission to access the basic user information from Company A, such as a user identifier or profile data. And the user can use the service provided by Company B with the account information provided by Company A without a separate account registration process.

This scenario enables:

- Redundant account registration during user onboarding.
- Clear separation between identity management and service ownership.
- Secure access to user information without sharing account credentials.

Scope and Limitations

This document provides a conceptual overview of OAuth-based authorization flows, focusing on how authorization enables controlled access to protected resources across independent systems, instead of diving into detailed OAuth protocol workflows.

The scope of this document is intentionally limited to system-level interactions and user-facing authorization scenarios. It is designed to support general understanding rather than serve as a protocol specification or implementation guide.

Scope

This document covers:

- The core roles involved in an OAuth workflow, including the resource owner, client application, authorization server, and resource server.
- High-level interaction patterns that depict how authorization requests, user consent, and resource access are organized across different roles.
- An end-to-end authorization flow that demonstrates the sequence of interactions required to obtain and use an access token.
- Some representative use cases that reflect common, real-world authorization scenarios.

Limitations

This document does not cover:

- Protocol-level specifications defined in OAuth protocol, including parameter definitions, endpoint details, and error handling.
- Implementation-specific considerations such as token formats, encryption mechanisms, or security configuration.
- Advanced OAuth extensions, including refresh tokens, or device-specific authorization flows.
- Identity and authentication process provided by OpenID Connect (OIDC), such as ID tokens and user identity assertions.

While some use cases described in this document may be implemented using OAuth in combination with additional standards (for example, OIDC for identity-related use cases), this document only focuses on the authorization model and intentionally abstracts identity concerns.