OAuth Working Group
Internet-Draft
Intended status: Informational
Expires: 1 November 2026

M. Chen
L. Su
China Mobile

30 April 2026

A Comprehensive Roadmap for OAuth 2.0 Standards and Drafts
draft-chen-oauth-roadmap-00

Abstract

The OAuth 2.0 ecosystem has expanded significantly since the publication of RFC 6749, resulting in a complex landscape of extensions, security best practices (BCPs), and application-specific profiles. This complexity can be daunting for implementers, architects, and security auditors. This document serves as a comprehensive roadmap to navigate this landscape. It categorizes key RFCs and active Internet-Drafts into functional areas, explains the relationships between them, and provides context on their evolution. The goal is to help readers understand the current state-of-the-art, select the appropriate specifications for their use cases, and follow the latest security best practices, while also offering a glimpse into the future directions of the framework.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 1 November 2026.

Copyright Notice

Table of Contents

1. Introduction

This document categorizes specifications within the OAuth ecosystem into three groups:

* *Published RFCs*: Form the cornerstone of current standards and best practices.

* *Active Drafts*: Represent features and future directions actively being developed by the working group.

* *Historical Drafts*: Are expired or superseded but provide valuable context for understanding the design decisions and technical evolution of OAuth.

This roadmap is intended to guide implementers, architects, and security professionals through the extensive library of OAuth specifications, helping them select the appropriate documents for their use cases.

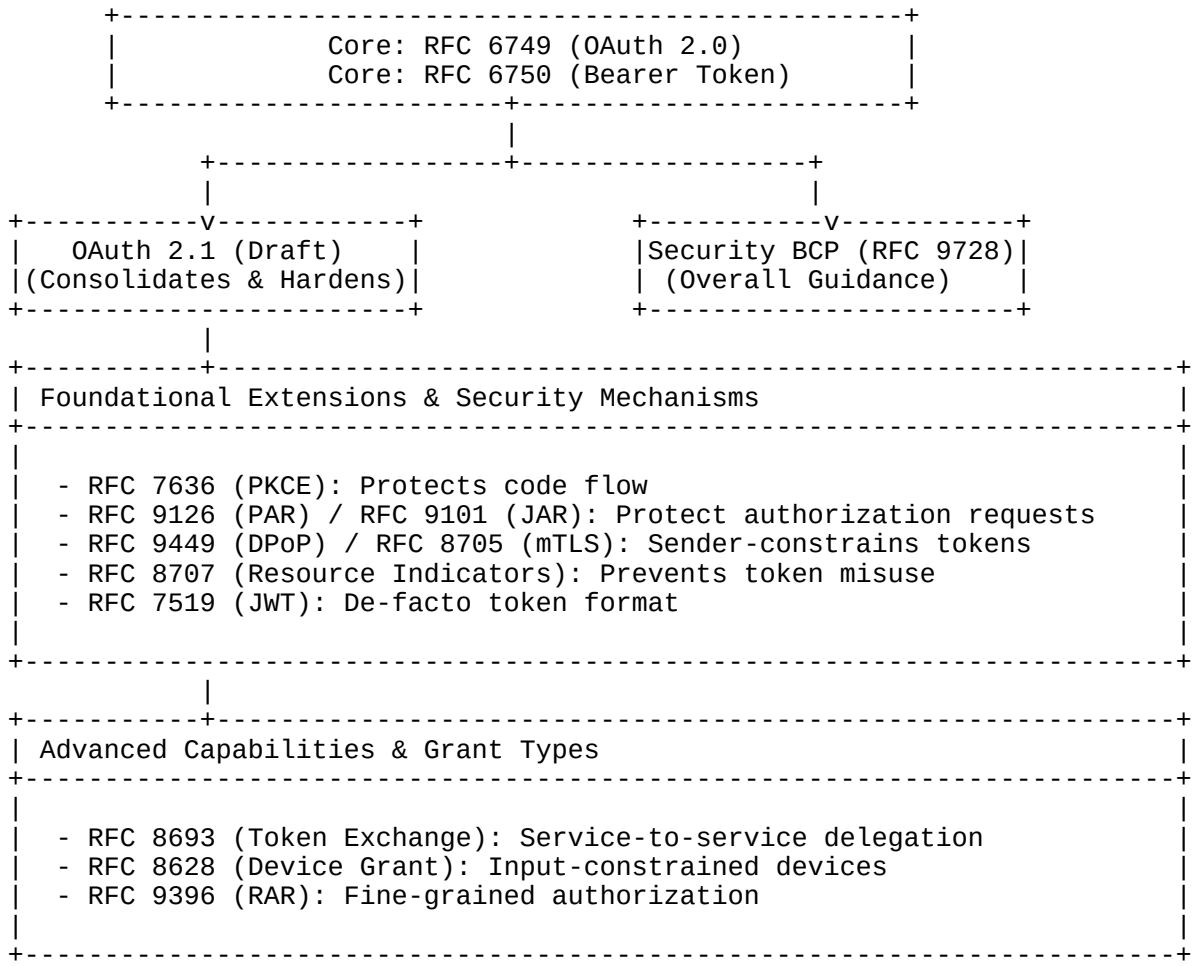

Figure 1: High-Level Relationship of Core OAuth 2.0 Specifications

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 RFC2119 [RFC8174].

Readers are expected to be familiar with the terms and concepts described in the core OAuth 2.0 Framework [RFC6749].

3. Core & Foundational Documents

* *RFC 6749*, The OAuth 2.0 Authorization Framework (S): The seminal specification for OAuth 2.0, defining the core roles (client, resource owner, authorization server, resource server), grant types, and authorization flows.

* *RFC 6750*, The OAuth 2.0 Authorization Framework: Bearer Token Usage (S): Defines the most common access token type, the "Bearer" token. A bearer token can be used by anyone who possesses it, making its transport and storage security critical.

* *draft-ietf-oauth-v2-1*, The OAuth 2.1 Authorization Framework (Draft): This effort consolidates and simplifies the original framework for modern applications. It formalizes a decade of security best practices by *mandating* PKCE, redirect URI exact matching, and the use of the Authorization Code flow. It explicitly *deprecates* insecure grants like the Implicit Grant and the Resource Owner Password Credentials Grant, providing a more secure baseline for new deployments.

4. Security Framework & Best Practices

4.1. Overall Security Guidance

* *RFC 6819*, OAuth 2.0 Threat Model and Security Considerations (I): An early comprehensive security document that analyzes various threat models (e.g., token leakage, open redirectors) and proposes countermeasures. While still useful, many of its recommendations have been superseded by RFC9728.

* *RFC 9728*, OAuth 2.0 Security Best Current Practice (B): The current definitive security guide. It consolidates the latest security lessons and provides a concrete set of recommendations that all modern implementations SHOULD follow.

* *draft-ietf-oauth-security-topics-update* (Draft): The draft name for the work that eventually became RFC9728, representing the continuous evolution of security guidance within the working group.

4.2. Protecting the Authorization Flow

* *RFC 7636*, Proof Key for Code Exchange (PKCE) (S): Protects the Authorization Code flow from interception attacks. *Originally designed for public clients (like mobile apps) that cannot securely store a secret, PKCE is now a BCP for all client types, including confidential ones*, as it provides a robust defense-in-depth against code injection.

* *RFC 9101*, JWT-Secured Authorization Request (JAR) (S): Ensures the integrity and authenticity of authorization request parameters by packaging them into a signed (and optionally encrypted) JWT. This prevents attackers from tampering with parameters like redirect_uri or scope.

* *RFC 9126*, Pushed Authorization Requests (PAR) (S): Enhances security and privacy by allowing a client to push its authorization request parameters directly to the authorization server via a secure backchannel. This returns a request_uri which the client then sends through the browser. *PAR protects request parameters from being exposed to the user agent (browser) or in server logs*, and it works very well in combination with JAR for a fully secured request.

4.3. Securing Access Tokens (Sender-Constraining)

* *RFC 8705*, Mutual-TLS Client Authentication and Certificate-Bound Access Tokens (S): A powerful mechanism that provides two functions: first, it defines a strict client authentication method using TLS certificates. Second, it enables the binding of access tokens to a specific client certificate, ensuring that even if a token is stolen, it cannot be used by an attacker who does not possess the client’s private key.

* *RFC 9449*, Demonstrating Proof-of-Possession (DPoP) (S): A lightweight, application-level sender-constraint mechanism. It binds a token to a specific client’s public/private key pair without relying on mutual-TLS. This makes it suitable for browser-based applications and other environments where mTLS is difficult to deploy.

* *draft-ietf-oauth-refresh-token-expiration* (Draft): Provides critical security best practices for refresh tokens, including recommendations for rotation (issuing a new refresh token with each use) and setting expiration policies to mitigate the risks of refresh token leakage.

4.4. Mitigating Logical & Implementation Flaws

* *RFC 8707*, Resource Indicators for OAuth 2.0 (S): Allows a client to specify the intended resource server (API) during the authorization request. This enables the Authorization Server to issue an audience-restricted token, preventing a token intended for one API from being replayed at another.

* *RFC 9207*, Authorization Server Issuer Identification (S): Prevents "mix-up" attacks where a malicious authorization server could trick a client into sending a code to the wrong token endpoint. It requires the AS to declare its issuer identifier and clients/RSs to validate it.

* *draft-ietf-oauth-mix-up-mitigation* (Draft): Provides comprehensive strategies to mitigate various "mix-up attacks" where a credential (e.g., a code, token, or client secret) might be sent to the wrong entity, expanding on the protections offered by RFC9207.

* *draft-ietf-oauth-cross-device-security* (Draft): Specifically analyzes and proposes mitigations for security threats in cross-device flows, such as the Device Authorization Grant (RFC8628).

4.5. Advanced Client Authentication

* *draft-ietf-oauth-attestation-based-client-auth* (Draft): Defines a client authentication method where the client must provide an attestation from a trusted third party (e.g., a device manufacturer or platform vendor), suitable for high-security scenarios where the client’s integrity must be verified.

* *draft-ietf-oauth-spiffe-client-auth* (Draft): Defines client authentication using SPIFFE SVIDs (Verifiable Identity Documents), designed to provide strong, cryptographic workload identity in cloud-native and service mesh environments.

5. Token Management & Formats

* *RFC 7009*, Token Revocation (S): Defines an endpoint for clients to proactively invalidate a refresh or access token, for example, when a user logs out.

* *RFC 7662*, Token Introspection (S): Defines an endpoint for resource servers to check the validity and metadata of a token with the authorization server. This is particularly useful for opaque (non-JWT) tokens.

* *RFC 7519*, JSON Web Token (JWT) (S): Defines the JWT format, the de facto standard for structured, self-contained tokens in the OAuth ecosystem.

* *RFC 8725*, JSON Web Token Best Current Practices (B): Provides essential security best practices for implementing and using JWTs, such as algorithm validation (alg header) and claim verification.

  - *draft-ietf-oauth-rfc8725bis* (Draft): An effort to update RFC8725 with new security advice and clarifications based on implementation experience.

* *RFC 9278*, JWT Profile for OAuth 2.0 Access Tokens (S): An official profile defining a recommended set of claims (like iss, exp, aud, client_id) for using a JWT as an OAuth 2.0 access token, promoting interoperability.

6. Client Registration, Metadata & Discovery

* *RFC 7591*, Dynamic Client Registration (S): Allows OAuth clients to register with an authorization server programmatically, automating the onboarding process.

* *RFC 7592*, Dynamic Client Registration Management (S): Complements RFC7591 by defining a protocol for updating and deleting client registrations.

* *RFC 8414*, Authorization Server Metadata (S): Allows an AS to publish its configuration details (like endpoint URLs and supported capabilities) at a well-known URI, enabling dynamic discovery by clients.

* *draft-ietf-oauth-client-id-metadata-document* (Draft): Proposes a mechanism for clients to also publish a metadata document, declaring their software information and configuration to enhance transparency and security for authorization servers.

7. Grant Types & Assertion Framework

* *RFC 7521*, Assertion Framework (S): Provides a general framework for using assertions (like SAML or JWT) for client authentication and as authorization grants. This is foundational for service-to-service communication.

* *RFC 7523*, JWT Profile for Authorization Grants (S): A concrete implementation of RFC7521 using JWTs as assertions. This allows a client to use a JWT to request an access token without a direct user interaction.

- *draft-ietf-oauth-rfc7523bis* (Draft): Aims to update RFC7523 to address ambiguities and align with modern security practices.

* *RFC 8628*, Device Authorization Grant (S): The "Device Flow," designed for input-constrained devices (like smart TVs or CLIs) that cannot host a web browser for user authentication.

* *RFC 8693*, Token Exchange (S): A versatile grant type that allows a service to exchange one type of security token for another. *This is the cornerstone of modern microservices security, enabling use cases like impersonation (a frontend service acting on behalf of a user) and delegation (a service calling another service with a more restricted token).*

* *draft-ietf-oauth-identity-assertion-authz-grant* (Draft): Proposes a new grant type that allows a client to use an identity assertion (like an ID Token from an external IdP) to obtain an access token, streamlining federated identity scenarios.

8. Advanced Authorization Capabilities

* *RFC 9396*, Rich Authorization Requests (RAR) (S): Extends OAuth to allow clients to request structured, fine-grained authorization (e.g., access to specific bank accounts with specific transaction limits) beyond simple string-based scopes.

* *RFC 9901*, Incremental Authorization for OAuth 2.0 (S): Allows a client to request additional permissions (scopes) from a user without forcing them to re-approve the permissions that were already granted.

* *draft-ietf-oauth-transaction-tokens* (Draft): Explores the use of short-lived tokens bound to a specific transaction to enhance security for high-risk operations like financial payments, building on ideas from RAR and sender-constraining.

9. Client-Specific Profiles & BCPs

* *RFC 8252*, OAuth 2.0 for Native Apps (B): Provides best current practices for implementing OAuth 2.0 in native mobile and desktop applications. It recommends using external user agents (like the system browser) and custom URI schemes or claimed "https" schemes for handling redirects.

* *draft-ietf-oauth-browser-based-apps* (Draft): The definitive guide for implementing OAuth 2.0 in Single-Page Applications (SPAs). It *recommends the Authorization Code flow with PKCE and

deprecates the Implicit Grant*. It also provides guidance on token storage (e.g., using backend-for-frontend patterns) and mitigating threats like Cross-Site Scripting (XSS).

* *draft-ietf-oauth-first-party-apps* (Draft): Offers specific security and implementation guidance for first-party applications, where the client and authorization server belong to the same entity, allowing for certain optimizations while maintaining security.

10. Relationship with OpenID Connect

While this document focuses on the OAuth 2.0 authorization framework, it is crucial to mention OpenID Connect (OIDC) [OIDC]. OIDC is a simple identity layer built on top of OAuth 2.0 that provides authentication and enables clients to verify the identity of the end-user. Many modern OAuth deployments use OIDC for user login and to obtain an ID Token, and then leverage standard OAuth mechanisms for API authorization using the Access Token. Understanding the distinction and synergy between OAuth (for "what a user can do") and OIDC (for "who the user is") is fundamental for many use cases.

11. Advanced & Emerging Technologies

Specifications in this area often intersect with decentralized identity, Verifiable Credentials (VCs), and privacy-enhancing technologies.

* *draft-ietf-oauth-sd-jwt-vc* (Draft): Defines how to carry Verifiable Credentials (VCs) using Selectively Disclosable JWTs (SD-JWTs), allowing users to disclose only necessary identity attributes during an OAuth flow, enhancing user privacy.

* *draft-ietf-oauth-status-list* (Draft): Proposes an efficient and privacy-preserving mechanism for checking the revocation status of credentials (like VCs) at scale.

* *draft-ietf-oauth-identity-chaining* (Draft): Explores a mechanism to securely pass and link multiple identity and authorization contexts during token exchange, forming a verifiable "identity chain." This is highly relevant for complex delegation scenarios in microservices.

* *draft-ietf-gnap-core-protocol* (Draft): Represents the next generation of authorization protocols, sometimes referred to as "OAuth 3.0" in spirit. GNAP (Grant Negotiation and Authorization Protocol) is a ground-up redesign aimed at addressing the complexities learned from OAuth 2.0. It features a unified

request/response structure, native support for sender-constraining, fine-grained authorization, and better separation of concerns. While not a direct extension of OAuth 2.0, it represents a major future direction for the IETF's work in this space.

12. Historical & Superseded Documents

These expired drafts are valuable for understanding the evolution of OAuth. They represent early explorations of ideas, some of which evolved into the standards we use today.

* *draft-ietf-oauth-authentication* / *draft-ietf-oauth-web-delegation*: Very early drafts that explored building an authentication layer on top of OAuth 2.0, serving as precursors to OpenID Connect.

* *draft-ietf-oauth-v2-http-mac*: Proposed a Message Authentication Code (MAC) token type as an alternative to Bearer tokens for stronger request signing, but it was not widely adopted due to complexity.

* *draft-ietf-oauth-pop-architecture* / *draft-ietf-oauth-pop-key-distribution*: Laid the early architectural foundation for "Proof-of-Possession" (PoP) tokens, whose core ideas evolved into the more mature mTLS (RFC8705) and DPoP (RFC9449) specifications.

* *draft-ietf-oauth-token-binding*: An early attempt at sender-constraint that aimed to bind OAuth tokens to the underlying TLS channel, but the underlying Token Binding protocol it depended on was not finalized by the IETF.

* *draft-ietf-oauth-closing-redirectors*: Specifically discussed and proposed solutions for mitigating the Open Redirector vulnerability, with its ideas being integrated into mainstream security best practices.

* *draft-ietf-oauth-use-cases*: An early document used to collect and articulate the use cases that shaped the design of OAuth 2.0.

* *draft-ietf-oauth-distributed* / *draft-ietf-oauth-reciprocal*: Explored authorization models for distributed and peer-to-peer scenarios, with some concepts finding more mature expression in specifications like Token Exchange (RFC8693).

13. IANA Considerations

This document has no IANA actions.

14. Security Considerations

This document is a roadmap that provides a guide to the OAuth 2.0 family of specifications; it does not define a protocol itself. As such, it does not introduce any new security considerations. The security of an OAuth 2.0 implementation depends on the proper application of the standards and best practices described in the referenced documents.

Readers are strongly encouraged to consult the security considerations sections of each individual RFC and draft. In particular, the following documents provide comprehensive security guidance and are considered essential reading for all implementers:

* *OAuth 2.0 Security Best Current Practice* [RFC9728]: The primary and most up-to-date guide for securing OAuth 2.0 deployments.

* *OAuth 2.0 for Native Apps* [RFC8252]: Essential security practices for mobile and desktop applications.

* *JSON Web Token (JWT) Best Current Practices* [RFC8725]: Critical guidance for anyone using JWTs as access tokens or in other contexts.

Implementers should pay close attention to the evolution of security drafts, such as draft-ietf-oauth-v2-1, which aim to simplify and harden the core framework.

15. Informative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <https://www.rfc-editor.org/rfc/rfc6749>.

[RFC9728] Jones, M.B., Hunt, P., and A. Parecki, "OAuth 2.0 Protected Resource Metadata", RFC 9728, DOI 10.17487/RFC9728, April 2025, <https://www.rfc-editor.org/rfc/rfc9728>.

[RFC8252] Denniss, W. and J. Bradley, "OAuth 2.0 for Native Apps", BCP 212, RFC 8252, DOI 10.17487/RFC8252, October 2017, <https://www.rfc-editor.org/rfc/rfc8252>.

[RFC8725] Sheffer, Y., Hardt, D., and M. Jones, "JSON Web Token Best Current Practices", BCP 225, RFC 8725, DOI 10.17487/RFC8725, February 2020, <https://www.rfc-editor.org/rfc/rfc8725>.

[OIDC] OpenID Foundation, "OpenID Connect", n.d., <https://openid.net/connect/>.

Authors' Addresses

Meiling Chen
China Mobile
BeiJing
China
Email: chenmeiling@chinamobile.com

Li Su
China Mobile
BeiJing
China
Email: suli@chinamobile.com