Deferred Token Response

draft-ietf-oauth-deferred-token-response-01

OAuth Working Group,

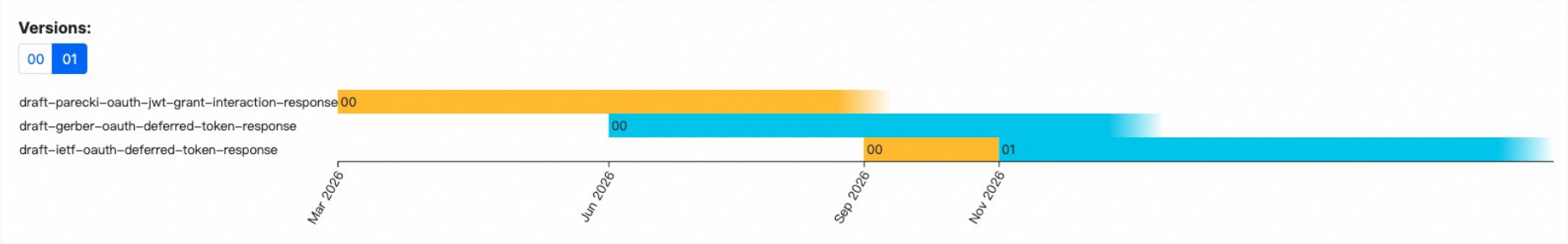
IETF 127, San Francisco

Wednesday 18 November 2026

Dapeng Liu



Changes since adoption (00 version): Integration with JAG-IR



- **Adopted**： August
  - 00 version: https://datatracker.ietf.org/doc/draft-ietf-oauth-deferred-token-response/00/

- **Integration with JAG-IR**: Nov
  - 01 version: https://datatracker.ietf.org/doc/draft-ietf-oauth-deferred-token-response/01/

https://github.com/oauth-wg/oauth-deferred-token-response

Changes since adoption (00 version): Add interactive mode

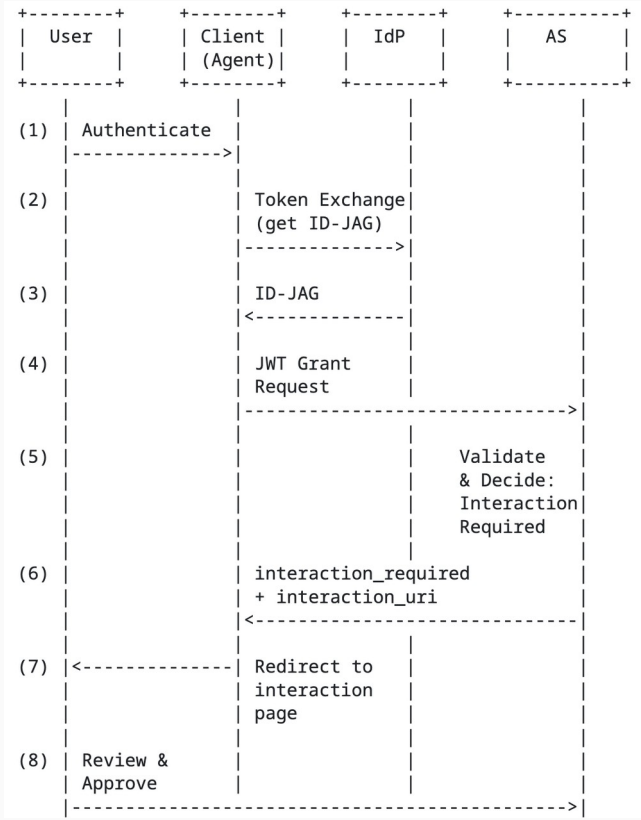
The draft defines the interactive flow:

- The ***interaction_required*** state and the ***interaction_uri*** response parameter (#49).
- The ***interaction_pending*** error code (#68) and its IANA registration (#77).
- The ***deferred_interactive*** completion mode (#71).
- The ***use cases*** covering the JAG-IR scenarios (#69).

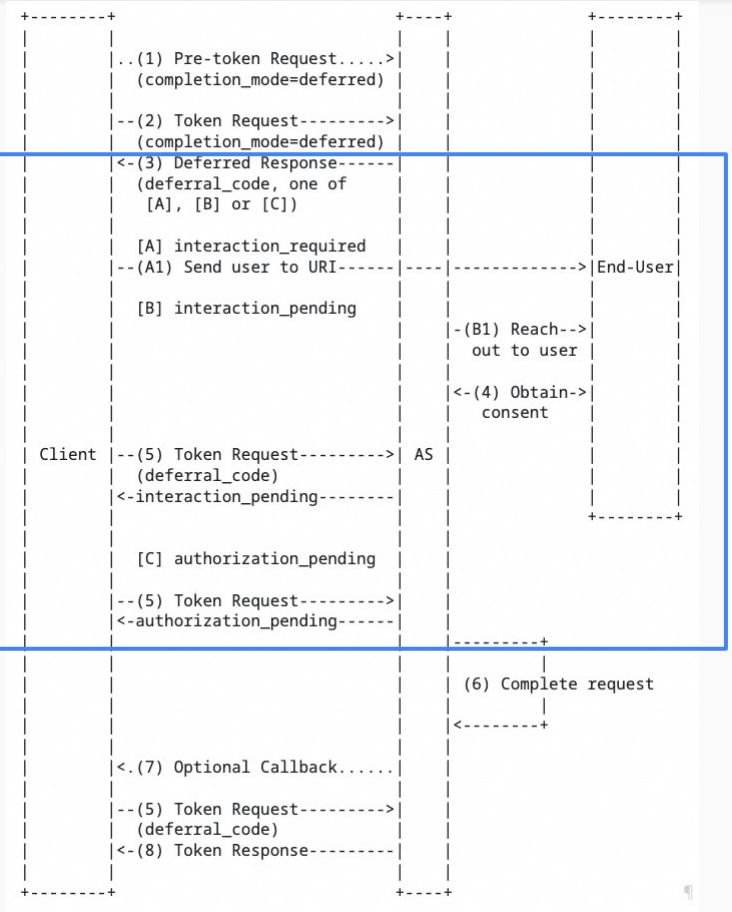
IETF

JAG-IR： JWT Authorization Grant Interaction Response

draft-parecki-oauth-jwt-grant-interaction-response-00



Changes since adoption (00 version): Updated the Overview section to cover the interaction states (#74).



Next Step：

- consider splitting the diagram into three different diagrams (issue#76)

Changes since adoption (00 version): Added a reference to RFC 9126 (PAR) (#78)

5.2. Pre-Token Hints

When the preceding request is an authorization request, completion_mode is conveyed like any other authorization request parameter. Examples include the request body of a Pushed Authorization Request (PAR)[RFC9126] or as a member of a JWT-secured authorization request object (JAR)[RFC9101].¶

IETF

Changes since adoption (00 version): Added implementation considerations on 128-bit entropy (#70)

6.3. Token Endpoint — Initial Request

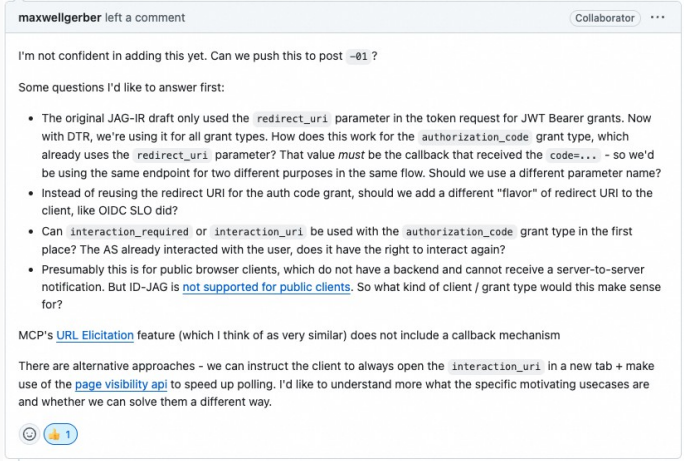
client_notification_token **OPTIONAL**. An opaque bearer credential generated by the client. If present, the authorization server **MUST** bind it to the resulting deferral code and **MUST** present it as a Bearer token on any callback notification for that deferral code (see Section 7.1). The token **MUST** contain sufficient entropy to make brute-force guessing infeasible (a minimum of 128 bits, with 160 bits **RECOMMENDED**), following Section 9.1 guidelines. A client that registers a deferred_client_notification_endpoint **SHOULD** include a client_notification_token; without one, the client cannot authenticate inbound callbacks (see Section 11.4).

IETF

Remaining issues: how redirect_uri should be handled in the JAG-IR flow (#79)

The PR adds ***redirect_uri*** to the token request, so that, after the user completes the interaction, the authorization server redirects the browser back to the client as a completion signal only.

The open question is: ***redirect_uri*** was originally designed for JAG-IR, and extending it to other grant types (e.g., **authorization_code, which already has a redirect_uri**) raises parameter compatibility issues that need more thought.



maxwellgerber left a comment — Collaborator

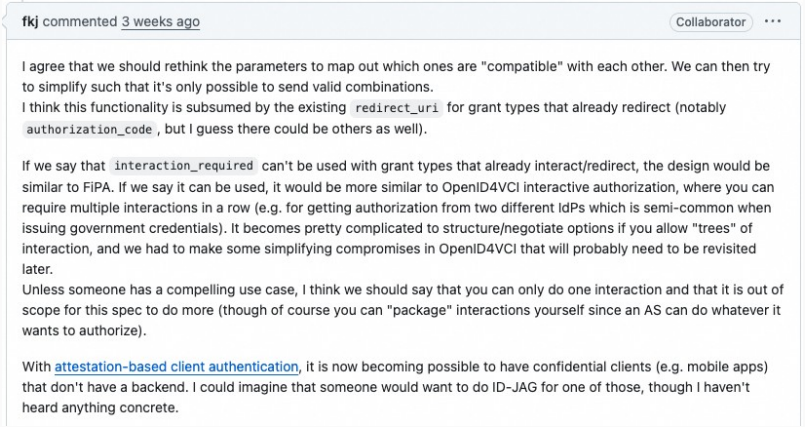
I'm not confident in adding this yet. Can we push this to post `-01`?

Some questions I'd like to answer first:

- The original JAG-IR draft only used the `redirect_uri` parameter in the token request for JWT Bearer grants. Now with DTR, we're using it for all grant types. How does this work for the `authorization_code` grant type, which already uses the `redirect_uri` parameter? That value *must* be the callback that received the `code=...` - so we'd be using the same endpoint for two different purposes in the same flow. Should we use a different parameter name?
- Instead of reusing the redirect URI for the auth code grant, should we add a different "flavor" of redirect URI to the client, like OIDC SLO did?
- Can `interaction_required` or `interaction_uri` be used with the `authorization_code` grant type in the first place? The AS already interacted with the user, does it have the right to interact again?
- Presumably this is for public browser clients, which do not have a backend and cannot receive a server-to-server notification. But ID-JAG is not supported for public clients. So what kind of client / grant type would this make sense for?

MCP's URL Elicitation feature (which I think of as very similar) does not include a callback mechanism

There are alternative approaches - we can instruct the client to always open the `interaction_uri` in a new tab + make use of the page visibility api to speed up polling. I'd like to understand more what the specific motivating usecases are and whether we can solve them a different way.



fkj commented 3 weeks ago — Collaborator

I agree that we should rethink the parameters to map out which ones are "compatible" with each other. We can then try to simplify such that it's only possible to send valid combinations.
I think this functionality is subsumed by the existing `redirect_uri` for grant types that already redirect (notably `authorization_code`, but I guess there could be others as well).

If we say that `interaction_required` can't be used with grant types that already interact/redirect, the design would be similar to FiPA. If we say it can be used, it would be more similar to OpenID4VCI interactive authorization, where you can require multiple interactions in a row (e.g. for getting authorization from two different IdPs which is semi-common when issuing government credentials). It becomes pretty complicated to structure/negotiate options if you allow "trees" of interaction, and we had to make some simplifying compromises in OpenID4VCI that will probably need to be revisited later.
Unless someone has a compelling use case, I think we should say that you can only do one interaction and that it is out of scope for this spec to do more (though of course you can "package" interactions yourself since an AS can do whatever it wants to authorize).

With attestation-based client authentication, it is now becoming possible to have confidential clients (e.g. mobile apps) that don't have a backend. I could imagine that someone would want to do ID-JAG for one of those, though I haven't heard anything concrete.

Feedback?

IETF

Remaining issues: Rehash use of deferral_code in Callback Notification #54

The Problem — Current Callback Design

- **Current Flow:** AS returns `deferral_code` on `/token`, then sends `deferral_code` **a second time** in the Callback Notification.

- **Security Flaws:**
  - **Double Exposure:** Transmits `deferral_code` over the wire twice.
  - **Plaintext Storage:** Forces AS to store `deferral_code` in plaintext instead of hashing it upon generation.

Proposed Resolution — Invert Token Generation

- **Core Change (Karl's Proposal):** Shift notification token generation from **Client** to **AS**.

- **Updated Mechanics:**
  1. **On Deferral (`/token` or `/authorize`):** AS generates **two separate artifacts**:
     - `deferral_code`: One-time secret for token redemption (**hashed** at rest on AS).
     - `notification_token` : Separate handle for callback routing.
  2. **On Callback Notification:** AS sends **only** the `notification_token` (or uses it as Bearer auth). `deferral_code` is **dropped completely** from callbacks.

Feedback?

IETF

Remaining issues: Allow the preceding endpoint to return a deferred response directly, in addition to accepting an advance hint #46

The Proposal (Direct Deferral at `/authorize`)

- **Core Idea:** Allows `/authorize` to return `deferral_code` directly when a decision is pending for Authorization Code Grant, rather than issuing a standard code and deferring at `/token`. —— ***Optional***
- **Benefits:** Prevents issuing a premature authorization code.

Discussion——which option?

- **Option A :** Keep core deferral restricted to the `/token` endpoint. —— ***No change***
- **Option B:** Adopt direct `/authorize` deferral with strict sender-constraining (`dpop_jkt`) [PR #47].

Feedback?

Other Comments: Mailing List Feedback

- **Token Issuance vs. Business Workflows:** Distinguishing when deferred token issuance is needed (cross-domain clients/agents) vs. resource-level risk enforcement (e.g., gateways / `ext_authz`).
- **Interaction Completion vs. Legal Consent:** Clarifying that completing an `interaction_uri` flow is not proof that legal requirements have been satisfied.
- **Scope Simplification:** Evaluating whether optional callbacks can be modularized to keep the core polling mechanism lean. [1]

Proposed Text Updates:

- Add an applicability and non-goals section to the overview to clarify the boundaries of asynchronous token issuance.
- Clarify that `interaction_uri` completion does not establish legal/contractual consent.
- Consider splitting Callbacks out into a separate optional Profile later on.

[1]: list discussion

Thanks!

IETF