Web Authorization Protocol
Internet-Draft
Intended status: Standards Track
Expires: 21 August 2026

Y. Zehavi
Raiffeisen Bank International
A. Parecki
Okta

17 February 2026

OAuth 2.0 direct interaction for native clients using federation
draft-zehavi-oauth-native-clients-federation-01

Abstract

OAuth 2.0 for First-Party Applications (FiPA) [I-D.ietf-oauth-first-party-apps] defined a native OAuth 2.0 *direct interaction*, whereby clients call authorization server's _Native Authorization Endpoint_ as an HTTP REST API, whose response instructs client what information to collect from end-user to satisfy authorization server's policies and requirements.

While FiPA [I-D.ietf-oauth-first-party-apps] focused on a one-to-one relationship between client and authorization server, this document is an *extension profile* adding support for authorization servers to federate the interaction to a downstream authorization server, instruct collection of additional information from users to guide request routing or instruct the usage of another native app for user interaction.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://yaron-zehavi.github.io/oauth-native-clients-federation/draft-zehavi-oauth-native-clients-federation.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-zehavi-oauth-native-clients-federation/.

Discussion of this document takes place on the Web Authorization Protocol Working Group mailing list (mailto:oauth@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/oauth/. Subscribe at https://www.ietf.org/mailman/listinfo/oauth/.

Source for this draft and an issue tracker can be found at https://github.com/yaron-zehavi/oauth-native-clients-federation.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document, OAuth 2.0 direct interaction for native clients using federation, extends FiPA [I-D.ietf-oauth-first-party-apps] to enable federation based flows, while retaining client's direct interaction with end-user.

The client calls the _Native Authorization Endpoint_ as an HTTP REST API, and receives instructions via the protocol established by FiPA, guiding client to interact with downstream authorization servers. This establishes a multi authorization server federated flow, whose user interactions are driven by the client app.

This document extends FiPA [I-D.ietf-oauth-first-party-apps] with new error responses: federate, redirect_to_app, insufficient_information and native_authorization_federate_unsupported.

It also adds additional response parameters: federation_uri, federation_body, response_uri, deep_link.

And adds the native_callback_uri request parameter.

2. Conventions and Definitions

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] when, and only when, they appear in all capitals, as shown here.

3. Protocol Overview

There are three primary ways this specification extends FiPA:

* federate response: Sends the client to interact with a downstream authorization server.

* insufficient_information response: Instructs the client to collect information from end-user required to decide where to federate to. For example this could be an email address which identifies the trust domain.

* redirect_to_app: Instructs the client to natively invoke an app to interact with end user.

3.1. Representative flow: Native client federated and redirected to app

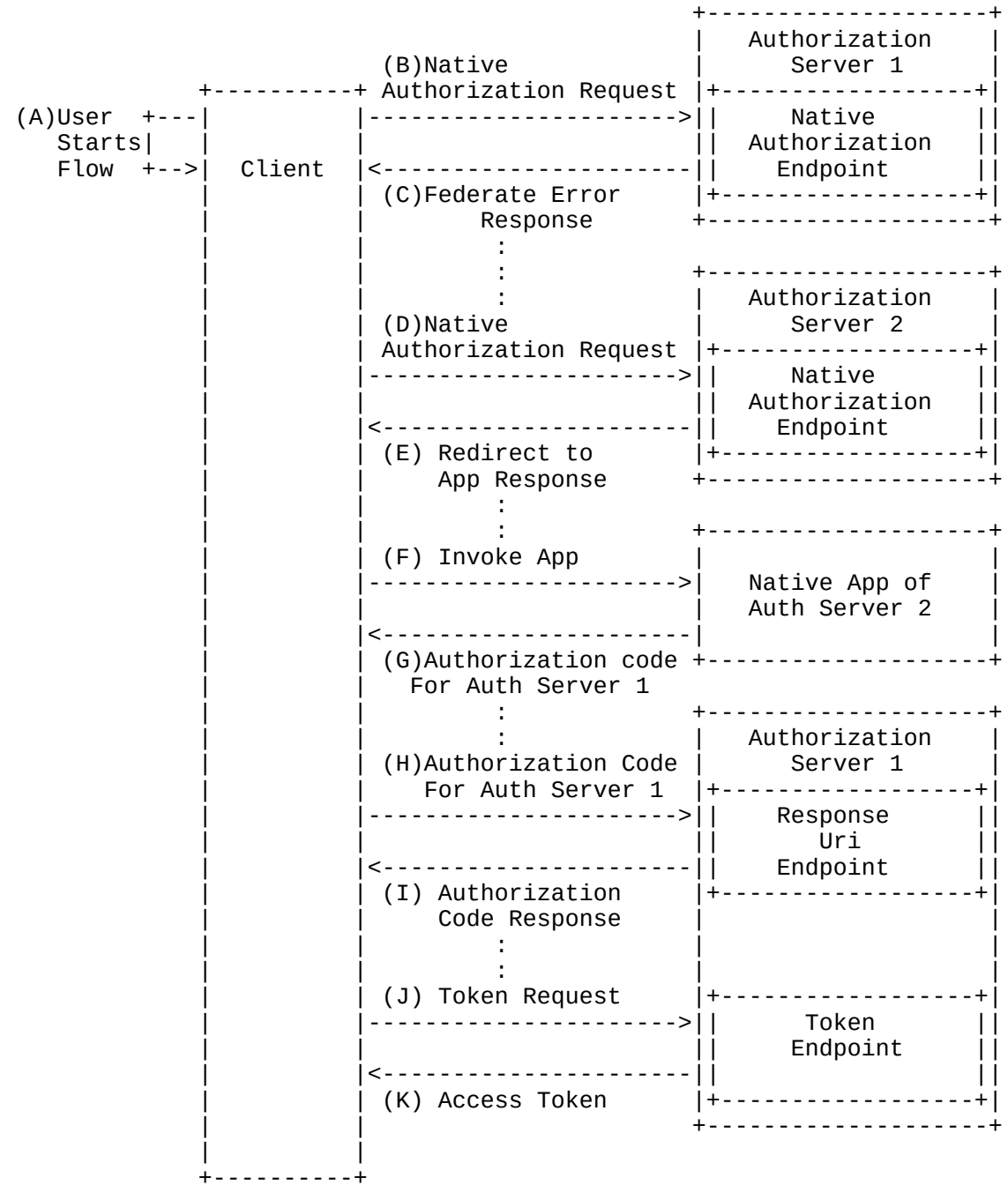

Figure: Native client federated, then redirected to app

* (A) The client starts the flow.

* (B) The client initiates the authorization request by making a POST request to the Native Authorization Endpoint of Authorization Server 1.

* (C) Authorization Server 1 decides to federate the user to Authorization Server 2. To do so it contacts Authorization Server 2's PAR [RFC9126] endpoint, then returns the federate error code together with the _federation_uri_, _federation_body_, _response_uri_ and _auth_session_ response attributes.

* (D) The client calls Authorization Server 2's Native Authorization Endpoint, as instructed by Authorization Server 1.

* (E) Authorization Server 2 decides to use a native app, and therefore responds with the redirect_to_app error code together with the _deep_link_ response attribute.

* (F) The client invokes the app using the deep_link.

* (G) The app interacts with user and if satisifed, returns an authorization code, regarded as Authorization Server 2's response to Authorization Server 1's federation to it.

* (H) The client provides the authorization code to Authorization Server 1.

* (I) Authorization Server 1 returns an authorization code.

* (J) The client sends the authorization code received in step (I) to obtain a token from the Token Endpoint.

* (K) Authorization Server 1 returns an Access Token from the Token Endpoint.

4. Protocol Endpoints

4.1. Native Authorization Endpoint

The native authorization endpoint defined by FiPA [I-D.ietf-oauth-first-party-apps] is used by this document.

This document adds the _native_callback_uri_ parameter to the native authorization endpoint, to support native user navigation across apps.

Before an authorization server instructs a client to federate to a downstream authorization server, it SHALL ensure the federated authorization server offers a _native_authorization_endpoint_, otherwise return the error _native_authorization_federate_unsupported_.

When federating to downstream authorization servers, the usage of PAR [RFC9126] with client authentication is REQUIRED, because when the client interacting with end-user calls the federated authorization server, it is not *its* OAuth client and therefore has no other means of authenticating. When using PAR with client authentication, the request_uri provided to the Native Authorization Endpoint attests that client authentication (by the federating authorization server) took place.

4.2. Native Authorization Request

The native authorization endpoint is called as defined by FiPA [I-D.ietf-oauth-first-party-apps]. This document adds the following request parameter:

"native_callback_uri": OPTIONAL. Native client app's *redirect_uri*, claimed as deep link. _native_callback_uri_ SHALL be natively invoked by authorization server's user-interacting app to provide its response to the client app. If native_callback_uri is included in a native authorization request, authorization server MUST include the native_callback_uri when federating to another authorization server.

4.3. Native Authorization Response

This document extends FiPA's [I-D.ietf-oauth-first-party-apps] error response, by adding the following error codes:

"error": REQUIRED. A single ASCII [USASCII] error code from the following:

- "insufficient_information": the Authorization Server requires additional user input, other than an authentication challenge, to determine the target authorization server to federate to. See Section 4.3.3 for details.

- "federate": The Authorization Server wishes to federate to another authorization server, which it is a client of. This response MUST include the _federation_uri_ response parameter. See Section 4.3.1 for details.

- "redirect_to_app": The Authorization Server wishes to fulfill the

user interaction using another native app. This response MUST include the _federation_uri_ response parameter. See Section 4.3.2 for details.

"native_authorization_federate_unsupported": The authorization server intended to federate to a downstream authorization server, but it does not support the native authorization endpoint.

And adds the following response attributes:

"federation_uri": OPTIONAL. The Native Authorization Endpoint of a downstream authorization server to federate to.

"deep_link": OPTIONAL. A URI of native app to be invoked to handle the request.

"federation_body": OPTIONAL. A string of application/x-www-form-urlencoded request parameters according to this specification for the downstream authorization server’s Native Authorization Endpoint.

"response_uri": OPTIONAL. A URI of an endpoint of federating authorization server which shall receive the response from the federated authorization server.

4.3.1. Federating Response

If the authorization server decides to federate to another authorization server, it responds with error code _federate_ and MUST return the _federation_uri_, _federation_body_, _response_uri_ and _auth_session_ response attributes.

When federating to another authorization server:

* Federating authorization server MUST use PAR [RFC9126] and include _request_uri_ in federation_body.

* If _native_callback_uri_ was included in the native authorization request, it MUST be included when calling federated authorization server’s Native Authorization Endpoint.

Example *federating* response:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{
    "error": "federate",
    "auth_session": "ce6772f5e07bc8361572f",
    "response_uri": "https://prev-as.com/native-authorization",
    "federation_uri": "https://next-as.com/native-authorization",
    "federation_body": "client_id=s6BhdRkqt3&request_uri=
      urn:ietf:params:oauth:request_uri:R3p_hzwsR7outNQSKfoX"
}
```

Client MUST call the _federation_uri_ using HTTP POST, and provide it _federation_body_ as application/x-www-form-urlencoded request body. Example:

```
POST /native-authorization HTTP/1.1
Host: next-as.com
Content-Type: application/x-www-form-urlencoded

client_id=s6BhdRkqt3&request_uri=
urn:ietf:params:oauth:request_uri:R3p_hzwsR7outNQSKfoX
```

The client MUST provide any response obtained from the *federated* authorization server, as application/x-www-form-urlencoded request body for the _response_uri_ of the respective *federating* authorization server which SHALL be invoked using HTTP POST.

However, when *federated* authorization server returns the following error codes: _federate_, _insufficient_authorization_, _insufficient_information_, _redirect_to_app_, _redirect_to_web_, client MUST handle these errors according to FiPA [I-D.ietf-oauth-first-party-apps] and this specification.

Example client calling receiving an authorization code response from the federated authorization server:

```
HTTP/1.1 200 OK
Server: next-as.com
Content-Type: application/json
Cache-Control: no-store

{
  "authorization_code": "uY29tL2F1dGhlbnRpY"
}
```

And providing it to the federating authorization server’s response_uri, adding previously obtained auth_session:

```
POST /native-authorization HTTP/1.1
Host: prev-as.com
Content-Type: application/x-www-form-urlencoded

authorization_code=uY29tL2F1dGhlbnRpY
&auth_session=ce6772f5e07bc8361572f
```

4.3.2. Redirect to app response

If the authorization server decides to nominate another native app to interact with end user, it responds with error code _redirect_to_app_ and MUST return the _deep_link_ response attribute.

Example *redirect_to_app* response:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{
    "error": "redirect_to_app",
    "deep_link": "https://next-as.com/native-authorization?
      client_id=s6BhdRkqt3&request_uri=
      urn:ietf:params:oauth:request_uri:R3p_hzwsR7outNQSKfoX"
}
```

Client MUST use OS mechanisms to invoke the obtained deep_link. If no app claiming deep_link is found on the device, client MUST terminate the flow and MAY attempt a normal non-native OAuth flow.

The invoked app handles the native authorization request:

* Validates the request and ensures it contains a _native_callback_uri_, Otherwise terminates the flow.

* Establishes trust in _native_callback_uri_ and validates that an app claiming it is on the device. Otherwise terminates the flow.

* Authenticates end-user and authorizes the request.

* Uses OS mechanisms to natively invoke _native_callback_uri_ and return to the client, providing it a response according to this specification’s response from a Native Authorization Endpoint, as url-encoded query parameters.

Note - trust establishment mechanisms in _native_callback_uri_ are out of scope of this specification. However we assume closed ecosystems could employ an allowList, and open ecosystems could leverage [OpenID.Federation]:

* Extract native_callback_uri’s DNS domain.

* Add the path /.well-known/openid-federation and perform trust chain resolution.

* Inspect client’s metadata for redirect_uri’s and validate *native_callback_uri* is included among them.

When the client is invoked on its native_callback_uri, it shall regard the invocation as a response from the authorization server which instructed _redirect_to_app_. Therefore, obtained response’s audience is the authorization server which federated the client to the authorization server which redirected the client to the app. See Section 4.3.1 for details.

Example URI used to invoke of client app on its claimed native_callback_uri:

https://client.example.com/cb?authorization_code=uY29tL2F1dGhlbnRpY

Example client invoking the response_uri *of the authorization server which federated it* to the authorization server, which redirected it to the app:

```
POST /native-authorization HTTP/1.1
Host: prev-as.com
Content-Type: application/x-www-form-urlencoded

authorization_code=uY29tL2F1dGhlbnRpY
&auth_session=ce6772f5e07bc8361572f
```

4.3.3. Additional Information Required Response

If additional user input is required, for example to determine where to federate to, the response body shall contain the following additional properties:

logo: OPTIONAL. URL or base64-encoded logo of _Authorization Server_, for branding purposes.

userPrompt: REQUIRED. A JSON object containing the prompt definition. The following parameters MAY be used:

* options: OPTIONAL. A JSON object that defines a dropdown/select input with various options to choose from. Each key is the parameter name to be sent in the response and each value defines the option:

- title: OPTIONAL. A string holding the input’s title.

- description: OPTIONAL. A string holding the input’s description.

- values: REQUIRED. A JSON object where each key is the selection value and each value holds display data for that value:

  - name: REQUIRED. A string holding the display name of the selection value.

  - logo: OPTIONAL. A string holding a URL or base64-encoded image for that selection value.

* inputs: OPTIONAL. A JSON object that defines an input field. Each key is the parameter name to be sent in the response and each value defines the input field:

  - title: OPTIONAL. A string holding the input’s title.

  - hint: OPTIONAL. A string holding the input’s hint that is displayed if the input is empty.

  - description: OPTIONAL. A string holding the input’s description.

Example of requesting end-user for 2 multiple-choice inputs:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{
    "error": "insufficient_information",
    "auth_session": "ce6772f5e07bc8361572f",
    "logo": "uri or base64-encoded logo of Authorization Server",
    "userPrompt": {
        "options": {
            "bank": {
                "title": "Bank",
                "description": "Choose your Bank",
                "values": {
                    "bankOfSomething": {
                        "name": "Bank of Something",
                        "logo": "uri or base64-encoded logo"
                    },
                    "firstBankOfCountry": {
                        "name": "First Bank of Country",
                        "logo": "uri or base64-encoded logo"
                    }
                }
            },
            "segment": {
                "title": "Customer Segment",
                "description": "Choose your Customer Segment",
                "values": {
                    "retail": "Retail",
                    "smb": "Small & Medium Businesses",
                    "corporate": "Corporate",
                    "ic": "Institutional Clients"
                }
            }
        }
    }
}
```

Example of requesting end-user for text input entry (email):

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{
    "error": "insufficient_information",
    "auth_session": "ce6772f5e07bc8361572f",
    "action": "prompt",
    "id": "request-identifier-2",
    "logo": "uri or base64-encoded logo of Authorization Server",
    "userPrompt": {
        "inputs": {
            "email": {
                "hint": "Enter your email address",
                "title": "E-Mail",
                "description": "Lorem Ipsum"
            }
        }
    }
}
```

The client gathers the required additional information and makes a POST request to the Native Authorization Endpoint. Example of response following end-user multiple-choice:

```
POST /native-authorization HTTP/1.1
Host: example.as.com
Content-Type: application/x-www-form-urlencoded

auth_session=ce6772f5e07bc8361572f
&bank=bankOfSomething
&segment=retail
```

Example of _Client App_ response following end-user input entry:

```
POST /native-authorization HTTP/1.1
Host: example.as.com
Content-Type: application/x-www-form-urlencoded

auth_session=ce6772f5e07bc8361572f
&email=end_user@example.as.com
```

5. Security Considerations

5.1. Non First-Party applications of federated authorization servers

A federated authorization server should consider end-user’s privacy and security to determine if it should present authorization challenges in federation scenarios. For example, it can label *federating* clients as such and avoid serving them (i.e: client’s interacting on their behalf) authorization challenges involving sensitive data, as these are not first party clients.

6. IANA Considerations

6.1. OAuth Parameters Registration

IANA has (TBD) registered the following values in the IANA "OAuth Parameters" registry of [IANA.oauth-parameters] established by [RFC6749].

Parameter name: native_callback_uri

Parameter usage location: Native Authorization Endpoint

Change Controller: IETF

Specification Document: Section 5.4 of this specification

7. Normative References

[I-D.ietf-oauth-first-party-apps]
Parecki, A., Fletcher, G., and P. Kasselman, "OAuth 2.0 for First-Party Applications", Work in Progress, Internet-Draft, draft-ietf-oauth-first-party-apps-02, 20 October 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-oauth-first-party-apps-02>.

[IANA.oauth-parameters]
IANA, "OAuth Parameters", <https://www.iana.org/assignments/oauth-parameters>.

[OpenID.Federation]
Hedberg, Ed, R., Jones, M. B., Solberg, A. A., Bradley, J., De Marco, G., and V. Dzhuvinov, "OpenID Federation 1.0", March 2025, <https://openid.net/specs/openid-federation-1_0.html>.

[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>.

[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>.

[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>.

[RFC9126] Lodderstedt, T., Campbell, B., Sakimura, N., Tonge, D., and F. Skokan, "OAuth 2.0 Pushed Authorization Requests", RFC 9126, DOI 10.17487/RFC9126, September 2021, <https://www.rfc-editor.org/rfc/rfc9126>.

[USASCII] Institute, A. N. S., "Coded Character Set -- 7-bit American Standard Code for Information Interchange, ANSI X3.4", 1986.

Appendix A. Example User Experiences

This section provides non-normative examples of how this specification may be used to support specific use cases.

A.1. Native client federated and redirected to app

A.1.1. Diagram

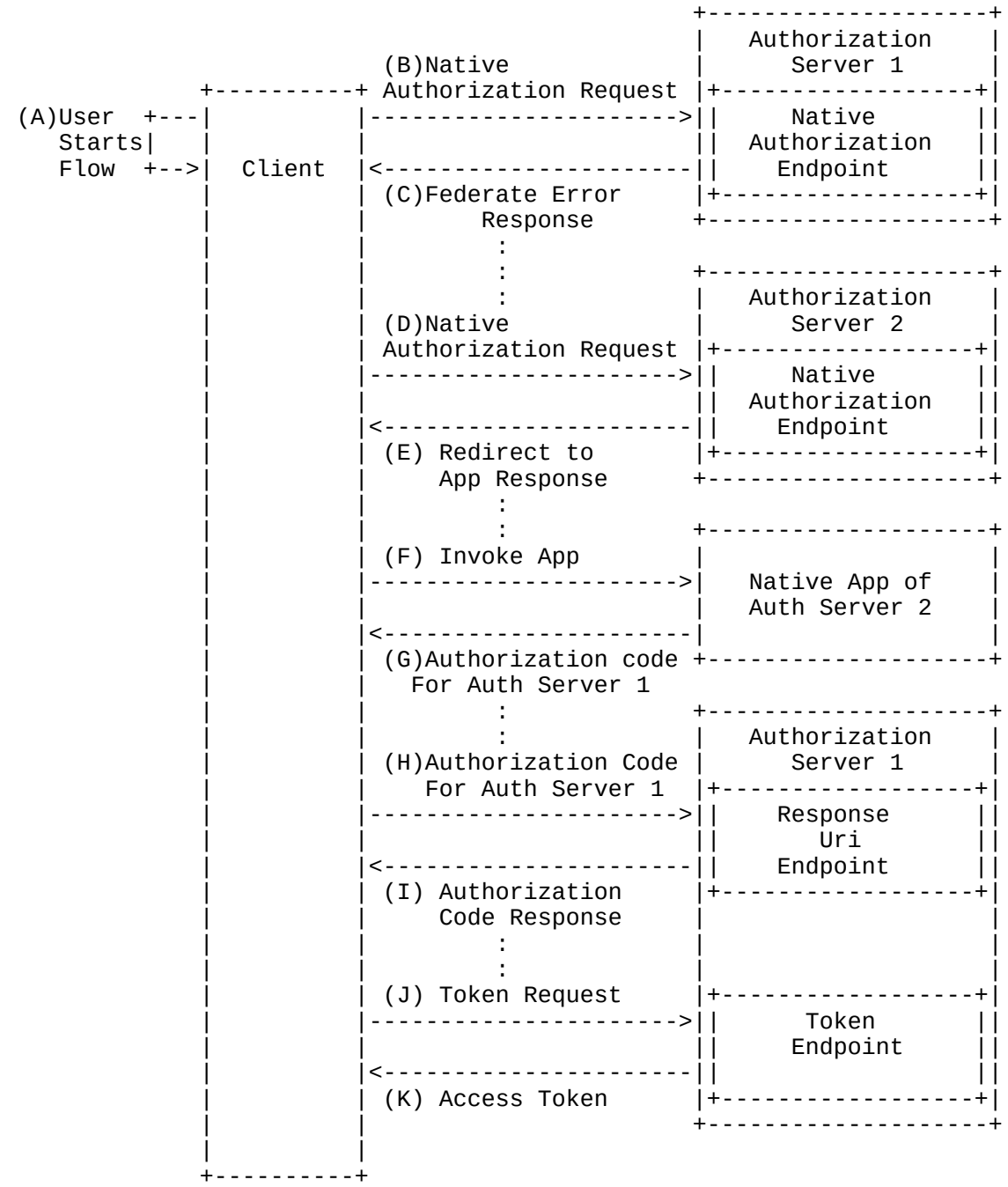

Figure: Native client federated, then redirected to app

A.1.2. Client makes initial request and receives "federate" error

Client calls the native authorization endpoint and includes the _native_callback_uri_ parameter:

```
POST /native-authorization HTTP/1.1
Host: as-1.com
Content-Type: application/x-www-form-urlencoded

client_id=t7CieSlru4&native_callback_uri=
https://client.example.com/cb
```

The first authorization server, as-1.com, decides to federate to as-2.com after validating it supports the native authorization endpoint. If it does not, as-1.com returns:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{
    "error": "native_authorization_federate_unsupported"
}
```

If native authorization endpoint is supported by the federated authorization server, as-1.com performs a PAR [RFC9126] request to as-2.com’s pushed authorization endpoint, including the original _native_callback_uri_:

```
POST /par HTTP/1.1
Host: as-2.com
Content-Type: application/x-www-form-urlencoded

client_id=s6BhdRkqt3&native_callback_uri=
https://client.example.com/cb
```

as-1.com receives a request_uri from as-2.com’s PAR endpoint, which it includes in its response to client, in the _federation_body_ attribute:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{
    "error": "federate",
    "auth_session": "ce6772f5e07bc8361572f",
    "response_uri": "https://as-1.com/native-authorization",
    "federation_uri": "https://as-2.com/native-authorization",
    "federation_body": "client_id=s6BhdRkqt3&request_uri=
      urn:ietf:params:oauth:request_uri:R3p_hzwsR7outNQSKfoX"
}
```

See Section 4.3.1 for more details.

A.1.3. Client calls federated authorization server and is redirected to app

Client calls the _federation_uri_ it got from as-1.com using HTTP POST with _federation_body_ as application/x-www-form-urlencoded request body:

```
POST /native-authorization HTTP/1.1
Host: as-2.com
Content-Type: application/x-www-form-urlencoded

client_id=s6BhdRkqt3&request_uri=
urn:ietf:params:oauth:request_uri:R3p_hzwsR7outNQSKfoX
```

as-2.com decides to use its native app to interact with end-user and responds:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{
    "error": "redirect_to_app",
    "deep_link": "https://as-2.com/native-authorization?
      client_id=s6BhdRkqt3&request_uri=
      urn:ietf:params:oauth:request_uri:R3p_hzwsR7outNQSKfoX"
}
```

Client locates an app claiming the obtained deep_link and invokes it. See Section 4.3.2 for more details. The invoked app handles the native authorization request and then natively invokes native_callback_uri:

```
https://client.example.com/cb?authorization_code=uY29tL2F1dGhlbnRpY
```

A.1.4. Client calls federating authorization server

Client invokes the response_uri of as-1.com as it is the authorization server which federated it to as-2.com and the app's response is regarded as the response of as-2.com:

```
POST /native-authorization HTTP/1.1
Host: as-1.com
Content-Type: application/x-www-form-urlencoded

authorization_code=uY29tL2F1dGhlbnRpY
&auth_session=ce6772f5e07bc8361572f
```

And receives in response an authorization code, which it is the audience of (no further federations) to resolve:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "authorization_code": "vZ3:uM3G2eHimcoSqjZ"
}
```

Client proceeds to exchange code for tokens.

Appendix B. Document History

-00

* Document creation

Acknowledgments

The authors would like to thank the attendees of IETFs 123 & 124 in which this was discussed, as well as the following individuals who contributed ideas, feedback, and wording that shaped and formed the final specification: George Fletcher, Arndt Schwenkshuster, Filip Skokan.

Authors' Addresses

Yaron Zehavi
Raiffeisen Bank International
Email: yaron.zehavi@rbinternational.com

Aaron Parecki
Okta

Email: aaron@parecki.com