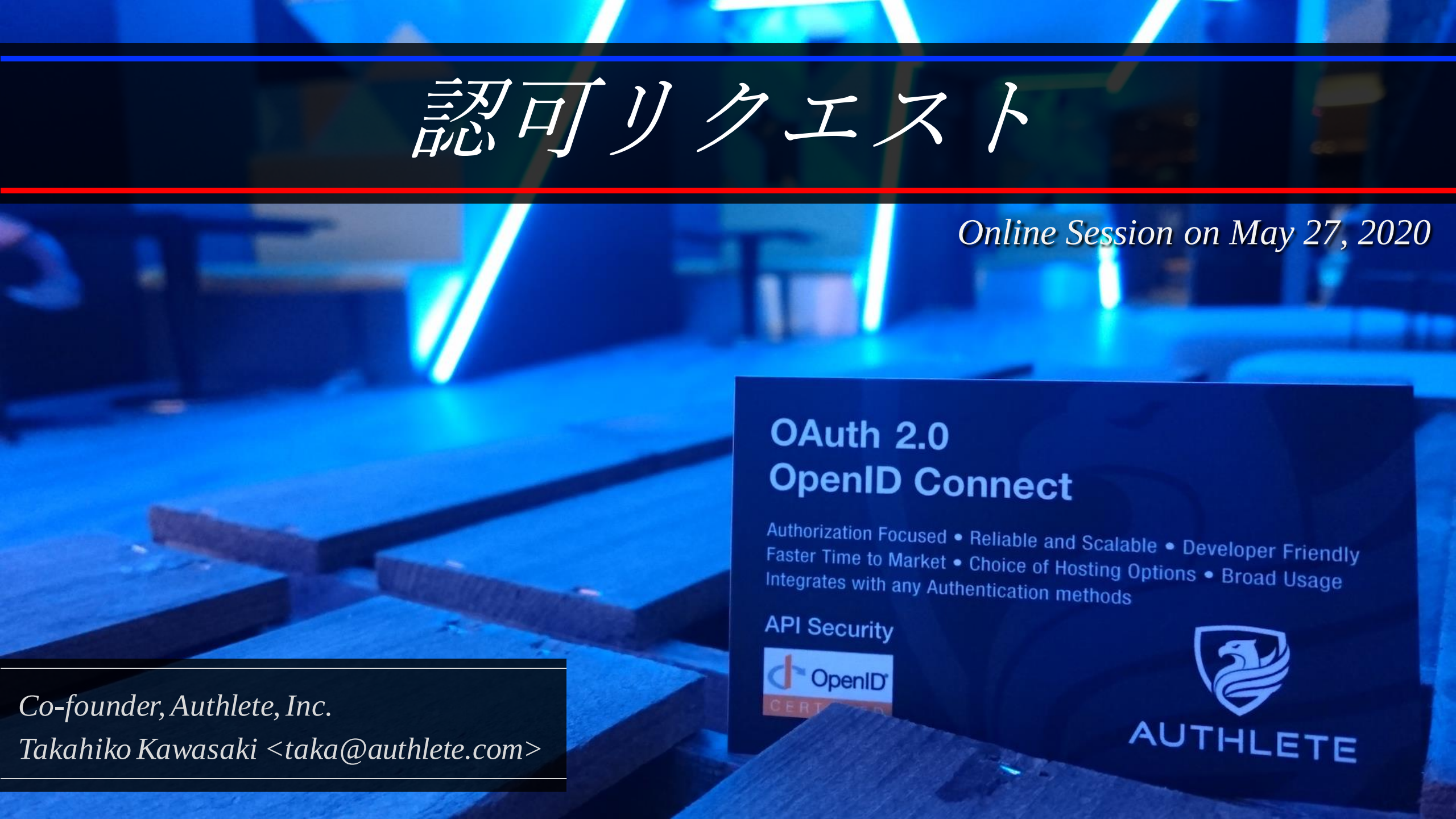
認可リクエスト

Online Session on May 27, 2020



Co-founder, Authlete, Inc.
Takahiko Kawasaki <taka@authlete.com>

前提知識

- RFC 6749 (The OAuth 2.0 Authorization Framework)
- RFC 7515 ～ RFC 7519 (JWS, JWE, JWK, JWA, JWT)
- RFC 7636 (PKCE)
- OpenID Connect Core 1.0, 2. ID Token
- OAuth 2.0 Multiple Response Type Encoding Practices

大前提知識

HTTP, HTML, JSON,
x-www-form-urlencoded,
Base64, Base64URL,
公開鍵暗号の概念

OAuth & OIDC 勉強会【入門編】（前提知識おさらい）

https://www.authlete.com/ja/resources/videos/20200317/

https://www.youtube.com/watch?v=PKPj_MmLq5E

勉強会資料を
ダウンロード！

OAuth & OIDC 勉強会【アクセストークン編】

https://www.authlete.com/ja/resources/videos/20200422/

https://www.youtube.com/watch?v=8WPPrzL-FQc&list=PLxDcFnLrbxvafUBu2GtX35G-iiktmZhWa

OpenID Connect Core 1.0, 3.1.2.1. Authentication Request

An Authentication Request is an OAuth 2.0 Authorization Request that requests that the End-User be authenticated by the Authorization Server.

→ 認証リクエストは、認可サーバーにエンドユーザー認証を要求する OAuth 2.0 認可リクエストである。

scope

REQUIRED. OpenID Connect requests MUST contain the `openid` scope value. If the `openid` scope value is not present, the behavior is entirely unspecified. （以下省略）

→ 必須。OpenID Connect リクエストは、`openid` スコープを含まなければならない。（以下省略）

使い分けるのも難しいので、以降全部『認可リクエスト』

※ 認可エンドポイントという呼び名は変わってないし

request parameter	specification
acr_values	OIDC Core 1.0 Section 3.1.2.1
authorization_details	OAuth 2.0 Rich Authorization Requests
claims	OIDC Core 1.0 Section 5.5
claims_locales	OIDC Core 1.0 Section 5.2
client_id	RFC 6749
code_challenge	RFC 7636
code_challenge_method	RFC 7636
display	OIDC Core 1.0 Section 3.1.2.1
id_token_hint	OIDC Core 1.0 Section 3.1.2.1
login_hint	OIDC Core 1.0 Section 3.1.2.1
max_age	OIDC Core 1.0 Section 3.1.2.1
nonce	OIDC Core 1.0 Section 3.1.2.1
prompt	OIDC Core 1.0 Section 3.1.2.1
redirect_uri	RFC 6749
registration	OIDC Core 1.0 Section 7.2.1
request	OIDC Core 1.0 Section 6
request_uri	OIDC Core 1.0 Section 6
resource	RFC 8707
response_mode	OAuth 2.0 Multiple Response Type Encoding Practices
response_type	RFC 6749
scope	RFC 6749
state	RFC 6749
ui_locales	OIDC Core 1.0 Section 3.1.2.1

code_challenge_method
plain
S256

display	
page	touch
popup	wap

prompt	
consent	none
login	select_account

response_mode	specification
form_post	OAuth 2.0 Form Post Response Mode
form_post.jwt	JWT Secured Authorization Response Mode for OAuth 2.0
fragment	OAuth 2.0 Multiple Response Type Encoding Practices
fragment.jwt	JWT Secured Authorization Response Mode for OAuth 2.0
jwt	JWT Secured Authorization Response Mode for OAuth 2.0
query	OAuth 2.0 Multiple Response Type Encoding Practices
query.jwt	JWT Secured Authorization Response Mode for OAuth 2.0

response_type	specification
code	RFC 6749
code id_token	OAuth 2.0 Multiple Response Type Encoding Practices
code id_token token	OAuth 2.0 Multiple Response Type Encoding Practices
code token	OAuth 2.0 Multiple Response Type Encoding Practices
id_token	OAuth 2.0 Multiple Response Type Encoding Practices
id_token token	OAuth 2.0 Multiple Response Type Encoding Practices
none	OAuth 2.0 Multiple Response Type Encoding Practices
token	RFC 6749

scope	specification
address	OIDC Core 1.0 Section 5.4
email	OIDC Core 1.0 Section 5.4
offline_access	OIDC Core 1.0 Section 11
openid	OIDC Core 1.0 Section 3.1.2.1
phone	OIDC Core 1.0 Section 5.4
profile	OIDC Core 1.0 Section 5.4

Request Object

OpenID Connect Core 1.0, Section 6. Passing Request Parameters as JWTs

request
: OPTIONAL. This parameter enables OpenID Connect requests to be passed in a single, self-contained parameter and to be optionally signed and/or encrypted. The parameter value is a Request Object value, as specified in Section 6.1. It represents the request as a JWT whose Claims are the request parameters.

request_uri
: OPTIONAL. This parameter enables OpenID Connect requests to be passed by reference, rather than by value. The `request_uri` value is a URL using the `https` scheme referencing a resource containing a Request Object value, which is a JWT containing the request parameters.

- 認可リクエストのパラメーター群を JWT にまとめて認可サーバーに送る方法
- リクエストのパラメーター群をまとめた JWT のことをリクエストオブジェクト（Request Object）と呼ぶ。

`request=`*`{JWT}`	リクエストオブジェクトを直接渡す (by value)
`request_uri=`*`{URI}`	リクエストオブジェクトの場所を渡す (by reference)

request を含む認可リクエストの例

```
https://server.example.com/authorize?
  response_type=code%20id_token
  &client_id=s6BhdRkqt3
  &redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb
  &scope=openid
  &state=af0ifjsldkj
  &nonce=n-0S6_WzA2Mj
  &request=eyJhbGciOiJSUzI1NiIsImtpZCI6ImsyYmRjIn0.ew0KICJpc3MiOiA
  iczZCaGRSa3F0MyIsDQogImF1ZCI6ICJodHRwczovL3NlcnZlci5leGFtcGxlLmN
  vbSIsDQogInJlc3BvbnNlX3R5cGUiOiAiY29kZSBpZF90b2tlbiIsDQogImNsaWV
  udF9pZCI6ICJzNkJoZFJrcXQzIiwNCiAicmVkaXJlY3RfdXJpIjogImh0dHBzOi8
  vY2xpZW50LmV4YW1wbGUub3JnL2NiIiwNCiAic2NvcGUiOiAib3BlbmlkIiwNCiA
  ic3RhdGUiOiAiYWYwaWZqc2xka2oiLA0KICJub25jZSI6ICJuLTBTNl9XekEyTWo
  iLA0KICJtYXhfYWdlIjogODY0MDAsDQogImNsYWltcyI6IA0KICB7DQogICAidXN
  lcmluZm8iOiANCiAgICB7DQogICAgICJnaXZlbl9uYW1lIjogeyJlc3NlbnRpYWw
  iOiB0cnVlfSwNCiAgICAgIm5pY2tuYW1lIjogbnVsbCwNCiAgICAgImVtYWlsIjo
  geyJlc3NlbnRpYWwiOiB0cnVlfSwNCiAgICAgImVtYWlsX3ZlcmlmaWVkIjogeyJ
  lc3NlbnRpYWwiOiB0cnVlfSwNCiAgICAgInBpY3R1cmUiOiBudWxsDQogICAgfSw
  NCiAgICJpZF90b2tlbiI6IA0KICAgIHsNCiAgICAgImdlbmRlciI6IG51bGwsDQo
  gICAgICJiaXJ0aGRhdGUiOiB7ImVzc2VudGlhbCI6IHRydWV9LA0KICAgICAiYWN
  yIjogeyJ2YWx1ZXMiOiBbInVybjptYWNlOmluY29tbW9uOmlhcDpzaWx2ZXIiXX0
  NCiAgICB9DQogIH0NCn0.nwwnNsk1-ZkbmnvsF6zTHm8CHERFMGQPhos-EJcaH4H
  h-sMgk8ePrGhw_trPYs8KQxsn6R9Emo_wHwajyFKzuMXZFSZ3p6Mb8dkxtVyjoy2
  GIzvuJT_u7PkY2t8QU9hjBcHs68PkgjDVTrG1uRTx0GxFbuPbj96tVuj11pTnmFC
  UR6IEOXKYr7iGOCRB3btfJhM0_AKQUfqKnRlrRscc8Kol-cSLWoYE9l5QqholImz
  jT_cMnNIznW9E7CDyWXTsO70xnB4SkG6pXfLSjLLlxmPGiyon_-Te111V8uE83Il
  zCYIb_NMXvtTIVc1jpspnTSD7xMbpL-2QgwUsAlMGzw
```

リクエストオブジェクト

OpenID Connect Core 1.0, Section 6.1.1. より抜粋

リクエストオブジェクトのペイロード部

```
{
 "iss": "s6BhdRkqt3",
 "aud": "https://server.example.com",
 "response_type": "code id_token",
 "client_id": "s6BhdRkqt3",
 "redirect_uri": " https://client.example.org/cb",
 "scope": "openid",
 "state": "af0ifjsldkj",
 "nonce": "n-0S6_WzA2Mj",
 "max_age": 86400,
 "claims":
  {
   "userinfo":
    {
     "given_name": {"essential": true},
     "nickname": null,
     "email": {"essential": true},
     "email_verified": {"essential": true},
     "picture": null
    },
   "id_token":
    {
     "gender": null,
     "birthdate": {"essential": true},
     "acr": {"values": ["urn:mace:incommon:iap:silver"]}
    }
  }
}
```

request_uri を含む認可リクエストの例

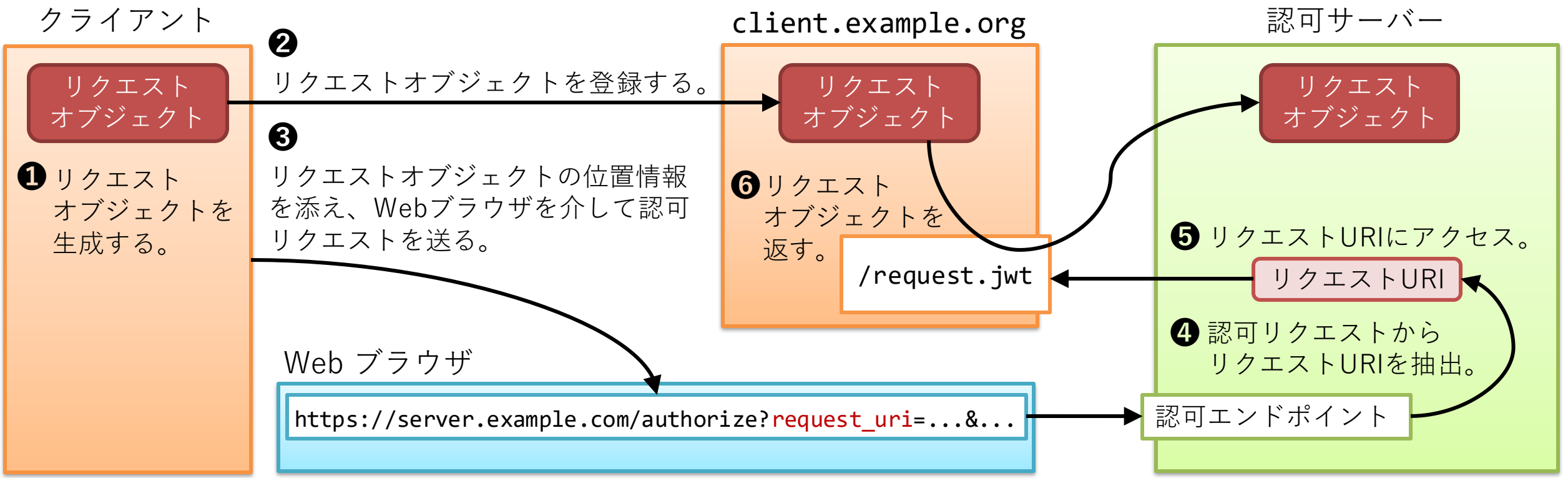
```
https://server.example.com/authorize?
  response_type=code%20id_token
  &client_id=s6BhdRkqt3
  &request_uri=https%3A%2F%2Fclient.example.org%2Frequest.jwt
  %23GkurKxf5T0Y-mnPFCHqWOMiZi4VS138cQO_V7PZHAdM
  &state=af0ifjsldkj&nonce=n-0S6_WzA2Mj
  &scope=openid
```

リクエストURI

OpenID Connect Core 1.0, Section 6.2.2. より抜粋

```
https://client.example.org/request.jwt#
  GkurKxf5T0Y-mnPFCHqWOMiZi4VS138cQO_V7PZHAdM
```

サーバーはリクエストURIが指すリソースをキャッシュしてもよい。もしもリソースの内容が変化する可能性があるなら、その内容のSHA-256ハッシュをbase64urlでエンコードした値をフラグメント部に含めるべき。



リクエストオブジェクト利用時の注意点 (OpenID Connect Core 1.0)

- ☑ `request` と `request_uri` を同時に使ってはならない。
- ☑ リクエストオブジェクト内のパラメーターが、外のものよりも優先される。
- ☑ 互換性のため、リクエストオブジェクトを使う場合も `response_type` と `client_id` は必須。リクエストオブジェクト内にも含まれている場合、値は一致しなければならない。
- ☑ リクエストオブジェクト内に `scope` が含まれていても、OpenID Connect リクエストであることを示すためには、`openid` を 含む `scope` が必須。
- ☑ リクエストオブジェクトが署名されている場合、`iss` と `aud` を含むべき。第三者が署名した場合を除き `iss` はクライアント ID、`aud` は OpenID Provider の発行者識別子であるべき。
- ☑ リクエストオブジェクトに署名と暗号化の両方が施されている場合、署名後に暗号化という順番でなければならない。
- ☑ リクエストオブジェクトに `request` と `request_uri` を含めてはならない。
- ☑ OpenID Provider はリクエスト URI の事前登録を要求することができる。
- ☑ リクエスト URI 全体で ASCII 512 文字をこえてはならない。
- ☑ 対象となるリクエストオブジェクトの署名検証が可能でない限り、リクエスト URI のスキームは `https` でなければならない。

リクエストオブジェクト利用時の注意点 (Financial-grade API Part 2)

- ✅ `request` と `request_uri` のどちらかを使わなければならない。
- ✅ 署名されていなければならない。
- ✅ 署名アルゴリズムは `PS256` もしくは `ES256` でなければならない。
- ✅ `exp` (Expiration Time) を含まなければならない。
- ✅ `aud` (Audience) を含まなければならない。
- ✅ リクエストオブジェクト外にあるリクエストパラメーター群は全て、リクエストオブジェクト内にも存在しなければならない。

The OAuth 2.0 Authorization Framework: JWT Secured Authorization Request (JAR)

https://datatracker.ietf.org/doc/draft-ietf-oauth-jwsreq/

- ⚠ リクエストオブジェクトを別仕様として切り出したもの。多少互換性を損なう部分がある。
- ✅ `client_id` および、`request` または `request_uri` のどちらか、が必須。他は任意。
- ✅ リクエストオブジェクトは署名されていなければならない。
- ✅ 参照するのはリクエストオブジェクト内のパラメーターのみ。

OpenID Connect Client Initiated Backchannel Authentication Flow – Core 1.0

- ☑ CIBA（シーバ）と呼ばれる仕様。RFC 6749 とは異なるフローを定義している。
- ☑ CIBA のフローは、認可エンドポイントへの認可リクエストではなく、バックチャネル認証エンドポイントへのバックチャネル認証リクエストから始まる。
- ☑ バックチャネル認証リクエストにも `request` というパラメーターがあり、これを使うことで、リクエストパラメーター群を JWT にまとめて送ることができる。（Section 7.1.1.）

CIBA のリクエストオブジェクトに関する要件は、OIDC のそれとはいろいろと異なる。
詳細は次の記事の『3.1. リクエストオブジェクト』を参照のこと。

世界最先端の認証認可技術、実装者による『CIBA』解説

https://qiita.com/TakahikoKawasaki/items/9b9616b999d4ce959ba3

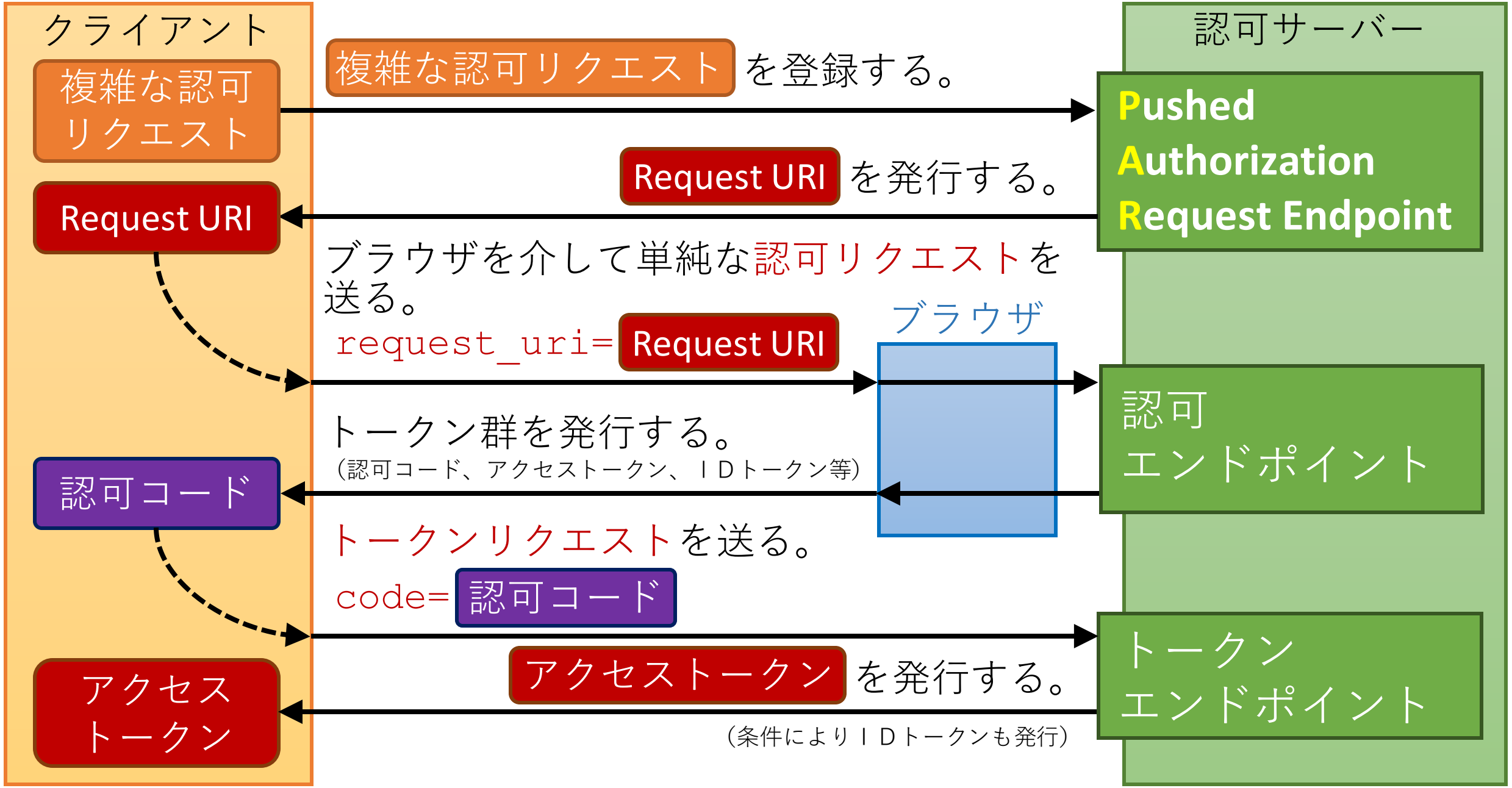
リクエストオブジェクトを置くサーバーを用意するのが面倒なので、
認可サーバーがその役割を担ってくれませんかね？（ﾁﾗｯ

つ *“Financial-grade API Part 2, 7. Request object endpoint”*

2019 年 12 月

切り出して、詳細詰めて標準化することにした。
“Pushed Authorization Requests for OAuth 2.0 (PAR)”

エンドポイントは *“Pushed Authorization Request Endpoint”* に
名称変更。汎用的に『認可リクエスト』を事前登録するので。


クライアント
認可サーバー
複雑な認可リクエスト
複雑な認可リクエスト を登録する。
Pushed Authorization Request Endpoint
Request URI を発行する。
Request URI
ブラウザを介して単純な認可リクエストを送る。
request_uri= Request URI
ブラウザ
認可エンドポイント
トークン群を発行する。
(認可コード、アクセストークン、ＩＤトークン等)
認可コード
トークンリクエストを送る。
code= 認可コード
アクセストークン を発行する。
アクセストークン
トークンエンドポイント
(条件によりＩＤトークンも発行)

PAR リクエストの例（PAR, Section 1）

```
POST /as/par HTTP/1.1
Host: as.example.com
Content-Type: application/x-www-form-urlencoded
Authorization: Basic czZCaGRSa3F0Mzo3RmpmcDBaQnIxS3REUmJuZlZkbUl3
                                                    ↑Client Type によってはクライアント認証が必要
response_type=code
&client_id=s6BhdRkqt3&state=af0ifjsldkj
&redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb
```

PAR レスポンスの例（PAR, Section 1）

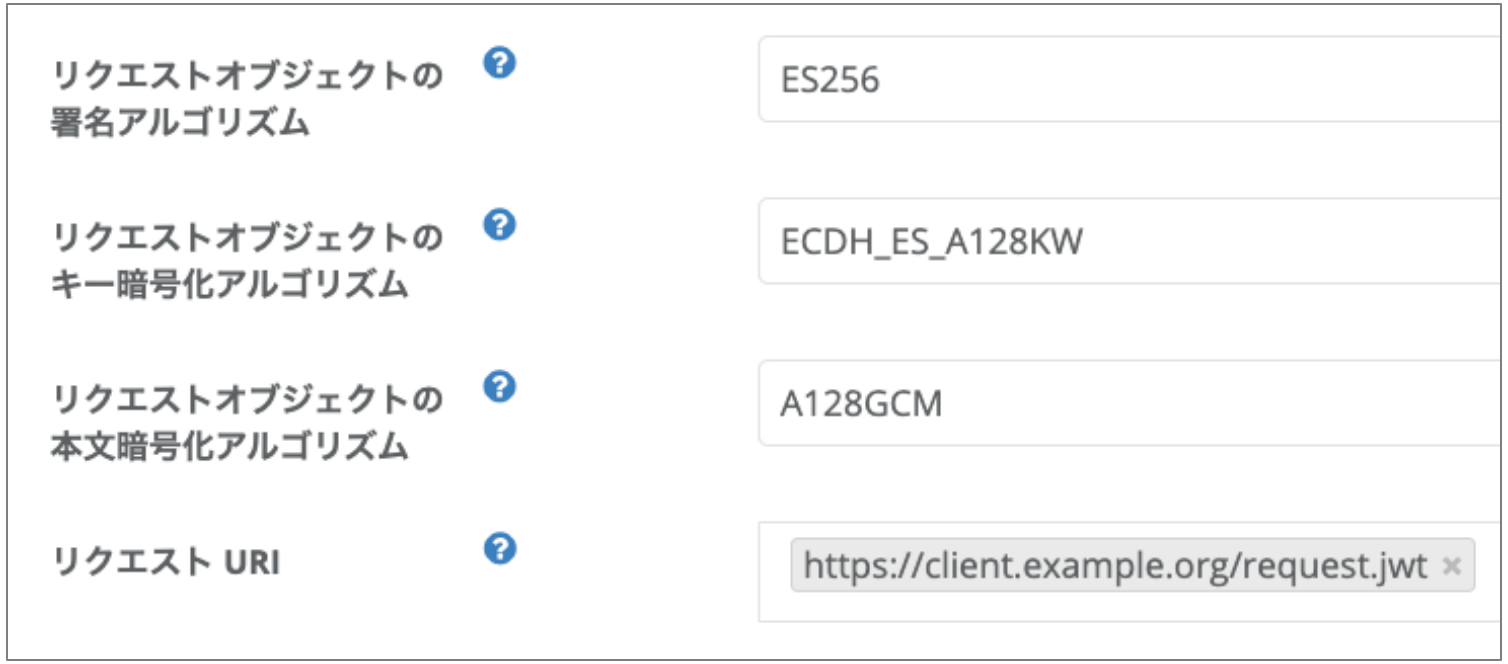
```
HTTP/1.1 201 Created
Cache-Control: no-cache, no-store
Content-Type: application/json

{                                                   ↓Request URI が発行されている
  "request_uri": "urn:example:bwc4JK-ESC0w8acc191e-Y1LTC2",
  "expires_in": 90
}
```

認可リクエストの例（PAR, Section 1）

```
GET /authorize?request_uri=                          ←Request URI を利用
  urn%3Aexample%3Abwc4JK-ESC0w8acc191e-Y1LTC2 HTTP/1.1
```

リクエストオブジェクト関連のクライアントメタデータ

request_object_signing_alg	リクエストオブジェクトの署名アルゴリズム
request_object_encryption_alg	リクエストオブジェクトの暗号アルゴリズム（鍵用）
request_object_encryption_enc	リクエストオブジェクトの暗号アルゴリズム（本文用）
request_uris	リクエスト URI のリスト



リクエストオブジェクトをサポートする認可サーバーのクライアント管理画面には上記メタデータに対応する設定項目がある。

← Authlete の管理コンソールの一部

リクエストオブジェクト関連のサーバーメタデータ

request_object_signing_alg_values_supported	サポートするリクエストオブジェクト署名アルゴリズム
request_object_encryption_alg_values_supported	サポートするリクエストオブジェクト鍵暗号アルゴリズム
request_object_encryption_enc_values_supported	サポートするリクエストオブジェクトの本文暗号アルゴリズム
request_parameter_supported	request パラメーターをサポートするか（真偽値）
request_uri_parameter_supported	request_uri パラメーターをサポートするか（真偽値）
require_request_uri_registration	リクエスト URI の事前登録を要求するか（真偽値）

OpenID Connect Core 1.0, Section 6.2. Passing a Request Object by Reference

> Note that Clients MAY pre-register request_uri values using the request_uris parameter defined in Section 2.1 of the OpenID Connect Dynamic Client Registration 1.0 [OpenID.Registration] specification. OPs can require that request_uri values used be pre-registered with the require_request_uri_registration discovery parameter.

→ require_request_uri_registration が true なら Request URI の事前登録が必要。

2017 年 7 月頃、OpenID Certification 取得に向けた作業を Justin Richer に依頼。しばらくして・・・

数多くの OAuth/OIDC 関連仕様の策定に携わっている有名な技術者。『OAuth 2 in Action』（邦訳『OAuth 徹底入門』）の著者。

justin

OpenID Certification のテストを Authlete に対して実行しているけど、「Request URI が登録されていない」という理由で失敗するね。

無登録の Request URI を許可したら、認可サーバーに任意の場所にネットワークアクセスさせるという攻撃が可能になり、セキュリティ上問題があるので、未登録の Request URI を拒否するように実装してあるからね。

`require_request_uri_registration` の値を `true` にしてあるのに、テストスイートはそれを見てくれていないのかな？

taka

justin

テストスイートのコードを見ると、メタデータの値に関係なく Request URI をランダムに生成し、認可サーバーに登録せずに使ってる。

ということは、無登録の Request URI も許可するようにしないと Certification 取れないね。無登録の Request URI の許可・不許可を選択するための設定項目を用意しようかな。

taka

justin

それは良くないと思う。セキュリティホールになってしまう。

taka

デフォルト設定を不許可にしておけば問題ないのでは？テストのときだけ許可する、と。

justin

いや、許可を選択可能にすること自体もダメ。

taka

そうはいっても、許可しないとテストをパスできなくて、Certification 取れないじゃん。

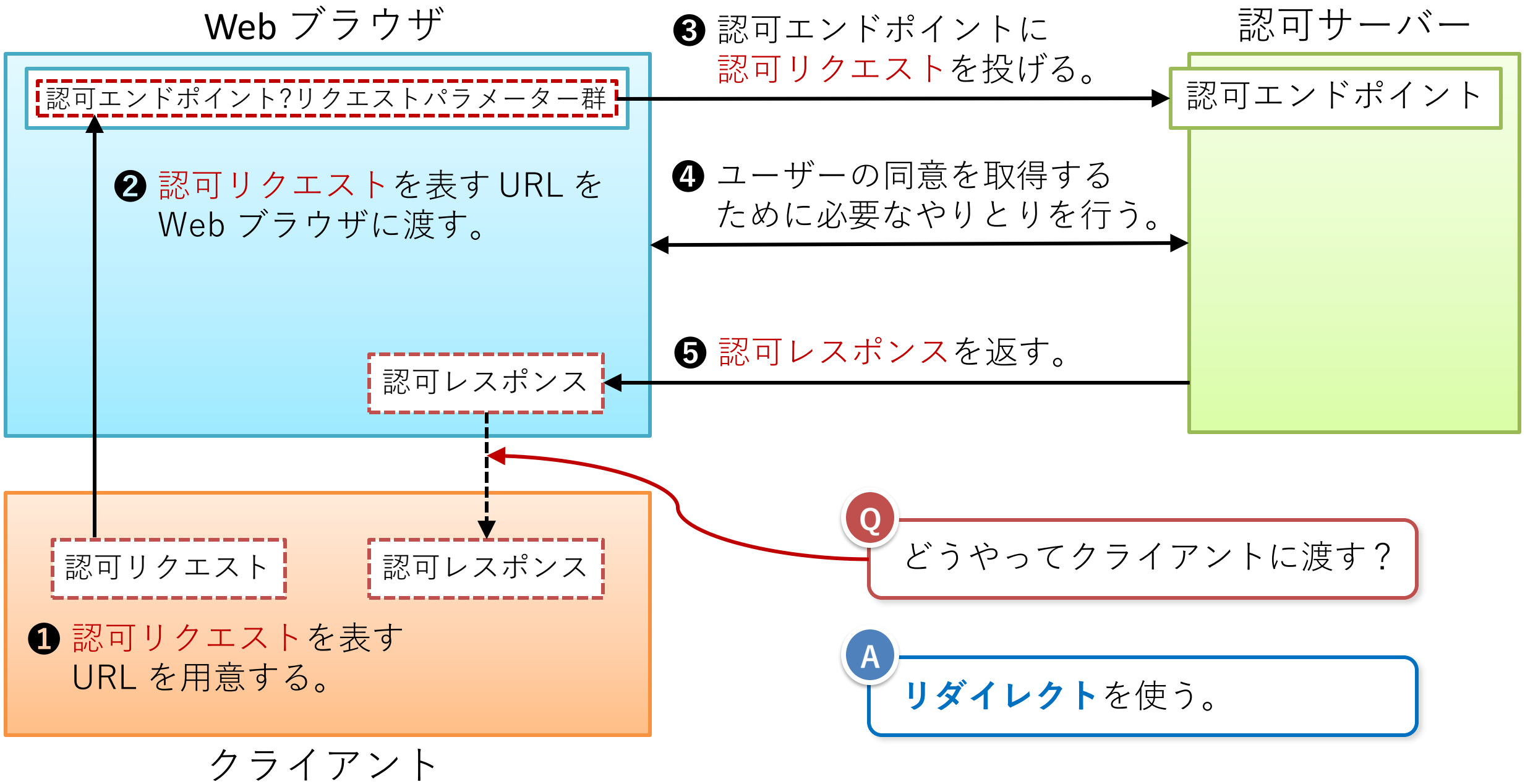
justin

僕に任せて。

この後 Justin は OpenID Foundation にかけあい、結果、テストスイート側が修正されることになった。

Justin さん、さすがですわ…

Redirection


Web ブラウザ
認可サーバー
認可エンドポイント?リクエストパラメーター群
❸ 認可エンドポイントに
認可リクエストを投げる。
認可エンドポイント
❷ 認可リクエストを表す URL を
Web ブラウザに渡す。
❹ ユーザーの同意を取得する
ために必要なやりとりを行う。
❺ 認可レスポンスを返す。
認可レスポンス
認可リクエスト
認可レスポンス
❶ 認可リクエストを表す
URL を用意する。
クライアント
Q
どうやってクライアントに渡す？
A
リダイレクトを使う。

- ✅ 一般的には、Web サーバーが成功応答を返す際は『200 OK』などの 200 番台の HTTP ステータスコードを用いる。

- ✅ 一方、認可サーバーが認可レスポンスを返す際は『302 Found』を用いる。
  （302 以外のケースについては後述）

- ✅ 『302 Found』には Location ヘッダーが伴う。

```
HTTP/1.1 302 Found
Location: 場所
```

- ✅ Web ブラウザは『302 Found』を受け取ると、Location ヘッダーが示す場所に遷移しようとする（＝その場所に対して HTTP リクエストを投げる）。

- ➡ 『302 Found』で Web ブラウザを別の場所に遷移させられる（リダイレクト）。

認可レスポンスの例

```
HTTP/1.1 302 Found
Location: https://example.com/callback?code=123&state=abc
```

Web ブラウザは Location に対応する HTTP リクエストを行う。

```
GET /callback?code=123&state=abc HTTP/1.1
Host: example.com
```

この HTTP リクエストの結果・・・

ホスト example.com の /callback で待ち受けているプログラムに
認可レスポンスパラメーター群 code=123&state=abc が渡る。

遷移先がクライアント開発者の管理下にあれば、認可レスポンスパラメーター群をクライアントに渡せる。

※ 遷移先のことを Redirection Endpoint と呼ぶ。

example.com の Web サーバー

/callback

❹ 認可レスポンスパラメーター群 `code=123&state=abc` を受け取る。

❸ 遷移先に HTTP リクエストが投げられる。

```
GET /callback?code=123&state=abc HTTP/1.1
Host: example.com
```

Web ブラウザ

`https://example.com/callback?code=123&state=abc`

❷ Web ブラウザは `Location` が示す場所に遷移する。

```
HTTP/1.1 302 Found
Location: https://example.com/callback
  ?code=123&state=abc
```

❶ 認可レスポンスを返す。

認可サーバー

インプリシットフローの場合

client.com の Web サーバー

/callback

❹ ここでは認可レスポンスパラメーター群を取得できない。

❸ 遷移先に HTTP リクエストが投げられるがフラグメント部は含まれない。

```
GET /callback HTTP/1.1
Host: example.com
```

❺ Web ブラウザに、JavaScript を含む HTML を送り返す。

```
HTTP/1.1 200 OK
Content-Type: text/html

<html>
......
<script src="client.js"></script>
</html>
```

Web ブラウザ

https://example.com/callback#access_token&token_type=Bearer&state=abc

❷ Web ブラウザは Location が示す場所に遷移する。

❻ JavaScript がフラグメント部に埋め込まれた認可レスポンスパラメーター群にアクセスする。

HTML & JavaScript

```
HTTP/1.1 302 Found
Location: https://client.com/callback
  #access_token=xyz&token_type=Bearer&state=abc
```

❶ 認可レスポンスを返す。

認可サーバー

Redirection URI

Redirection URI に関する要求事項（RFC 6749, Section 3.1.2）

- ☑ 絶対 URI（RFC 3986, 4.3 Absolute URI）でなければならない。
- ☑ クエリー部を含んでもよい。
- ☑ フラグメント部を含んではならない。

追加要求事項（OIDC Dynamic Client Registration 1.0, Section 2, `application_type`）

- ☑ 条件：`application_type` が `web` で、インプリシットフローを使うクライアント
  → スキーム部は `https` のみ。`localhost` をホスト部に使ってはならない。
- ☑ 条件：`application_type` が `native` のクライアント
  → スキーム部がカスタム、もしくは、スキーム部が `http` でホストが `localhost` 。

追加要求事項（BCP 212 (OAuth 2.0 for Native Apps), Section 7.1）

- ☑ 条件：スキーム部がカスタム
  → スキームはドメイン名を逆順にしたものでなければならない。（RFC 7595, Section 3.8）

追加要求事項（Financial-grade API Part 1, Section 5.2.2, 20）

- ☑ スキーム部は `https` でなければならない。（BCP 212, 7.2. Claimed "https" Scheme URI Redirection）

OpenID Connect Dynamic Client Registration 1.0, Section 2. Client Metadata

application_type

OPTIONAL. Kind of the application. The default, if omitted, is `web`. The defined values are `native` or `web`. Web Clients using the OAuth Implicit Grant Type MUST only register URLs using the `https` scheme as `redirect_uris`; they MUST NOT use `localhost` as the hostname. Native Clients MUST only register `redirect_uris` using custom URI schemes or URLs using the `http:` scheme with `localhost` as the hostname. （以下省略）

- `application_type` 省略時は `web`。そのため、必ず `native` か `web` のどちらかになる。
  - どのクライアントにも Redirection URI に関する上記追加要求事項が適用される。
    - 制約が強いので、全ての OAuth デプロイメントに適用すべきか確信が持てない。
      - Authlete の実装
        - ☑ `applicationType` は、`WEB`、`NATIVE` の他、「設定しない」を選べる。
        - ☑ Redirection URI に問題があれば、実行時（認可リクエスト時）にはじく。

Redirection URI の登録に関する要求事項（RFC 6749, Section 3.1.2.2）

- ☑ パブリッククライアントは登録必須。
- ☑ インプリシットフローを使うコンフィデンシャルクライアントは登録必須。
  - → 逆に言うと、インプリシットフロー以外を使うコンフィデンシャルクライアントは登録不要。
  - → とは言え、認可サーバーは全てのクライアントに登録を要求すべき（SHOULD）とされている。
- ☑ 複数の Redirection URI を登録してもよい。

追加要求事項（OIDC Core 1.0, Section 3.1.2.1, `redirect_uri`）

- ☑ 登録必須。OpenID Connect の Authentication Request では `redirect_uri` パラメーターが必須であり、その値が登録済みの URI のいずれかとマッチしなければならないと書かれているため。
  - → Financial-grade API など、OIDC Core をベースにしているものは、同様に登録必須となる。

追加要求事項（OIDC Dynamic Client Registration 1.0, Section 2, `redirect_uris`）

- ☑ 登録必須。`redirect_uris` は 必須であり、認可リクエストの `redirect_uri` パラメーターで指定される URI が登録済みの URI のいずれかとマッチしなければならないと書かれているため。

Redirection URI の検証に関する要求事項（OpenID Connect）

OpenID Connect の場合は、むしろ検証が楽。なぜなら・・・

- ☑ 認可リクエストに `redirect_uri` パラメーターが必須。
- ☑ Redirection URI の事前登録が必須。
- ☑ Redirection URI の比較は Simple String Comparison（RFC 3986, 6.2.1）で行う。

Redirect URI 登録・更新時に次のチェックをおこなっておけば、

- ☑ RFC 6749 の要求事項（フラグメント部を含まない絶対 URI）に準拠している

か

実行時の Redirect URI 検証は次のチェックだけで済む。

- ☑ 認可リクエストが `redirect_uri` パラメーターを含んでおり、その値が事前登録されている Redirection URI 群のいずれかと完全一致する。
- ☑ Application Type に基づく追加検証。（登録・更新時にこの検証を行う実装も当然ありうる）
- ☑ トークンリクエストが `redirect_uri` パラメーターを含んでおり、その値が認可リクエスト時に指定された値と完全一致する。（これ自体は RFC 6749, 4.1.3. の要求事項）

Redirection URI の検証に関する要求事項（RFC 6749）

RFC 6749 の場合は、むしろ検証が面倒。なぜなら・・・

- ☑ 認可リクエストの `redirect_uri` パラメーターはオプショナルだが条件により必須。
- ☑ Redirection URI の事前登録は条件によっては不要。
- ☑ Redirection URI の比較は正規化してから行うが（RFC 3986, 6）、条件により Simple String Comparison（RFC 3986, 6.2.1）。

実装上面倒なポイントの例

コンフィデンシャルクライアントが認可コードフローを使う場合、Redirection URI の事前登録は不要である。しかし、事前登録していないならば、認可リクエスト時に `redirect_uri` パラメーターが必須となる。このケースで指定される Redirection URI は事前登録されたものではないから、Redirection URI の検証は全て実行時におこなうことになる。

→ Authlete の実装が Redirection URI の検証作業を実行時に寄せている理由の一つ。

『OAuth 2.0 + OpenID Connect のフルスクラッチ実装者が知見を語る』 / リダイレクト URI

`https://qiita.com/TakahikoKawasaki/items/f2a0d25a4f05790b3baa`

Redirection URI の検証に関する要求事項（Financial-grade API）

Financial-grade API（FAPI）の場合、OpenID Connect の要求事項に加えて、

- ☑ Redirection URI のスキームが `https` であることを確認する。

Q

認可エンドポイントの実装は、(a) RFC 6749、(b) OIDC、(c) FAPI、の内の
どの要求事項に基づいて検証をおこなうべきか、どうやって判断するのか？

A

【解1：静的】設定固定や製品ライン分割等により、決め打ちで対応する。

【解2：動的】認可リクエストの内容から、適用すべき要求事項を決める。

→ 具体的にどうやるの？

RFC 6749 か OIDC かの動的判定

OpenID Connect Core 1.0, Section 3.1.2.1. Authentication Request

> scope
>
> REQUIRED. OpenID Connect requests MUST contain the `openid` scope value. If the `openid` scope value is not present, the behavior is entirely unspecified. （以下省略）

→ `scope` パラメーターに `openid` が含まれていれば OIDC リクエストとみなす。

FAPI かどうかの動的判定

→仕様内に動的判定に使えそうな記述が存在しないので、個々のベンダーが独自の仕組みを実装することになる。

→とは言え、論理的に詰めていくと、`scope` の内容に基づき、FAPI Part 1（Read-Only）、FAPI Part 2（Read-Write）、非 FAPI、の区別をする方法に行き着く。

Authlete の実装

Authlete FAPI Enhancements (Part 2)
https://www.authlete.com/ja/resources/videos/20180724/authlete-fapi-enhancements_2/
『Financial APIs Workshop 2018』でのプレゼン動画と、その文字起こし

1. 任意の Key-Value 属性群をスコープに紐付ける汎用的な仕組みを実装。
2. 属性名 `fapi` を特別扱いし、その値が `r` なら FAPI Read-Only、`rw` なら FAPI Read-Write と解釈する。
3. 認可リクエストに含まれるスコープ群を調べ、それらの一つでも `fapi=rw` という属性を持っていれば FAPI Read-Write リクエストとして扱う。
4. そうではない場合、`fapi=r` という属性を持つスコープが一つでもあれば、FAPI Read-Only リクエストとして扱う。
5. そうではない場合、FAPI リクエストとはみなさない。

app2app


OAuth 2.0
OpenID Connect
Authorization Focused • Reliable and Scalable • Developer Friendly
Faster Time to Market • Choice of Hosting Options • Broad Usage
Integrates with any Authentication methods
API Security
OpenID
AUTHLETE

Implementing app-to-app authorisation in OAuth2 / OpenID Connect

https://josephheenan.blogspot.com/2019/08/implementing-app-to-app-authorisation.html

Joseph Heenan (ジョセフ ヒーナン) @josephheenan

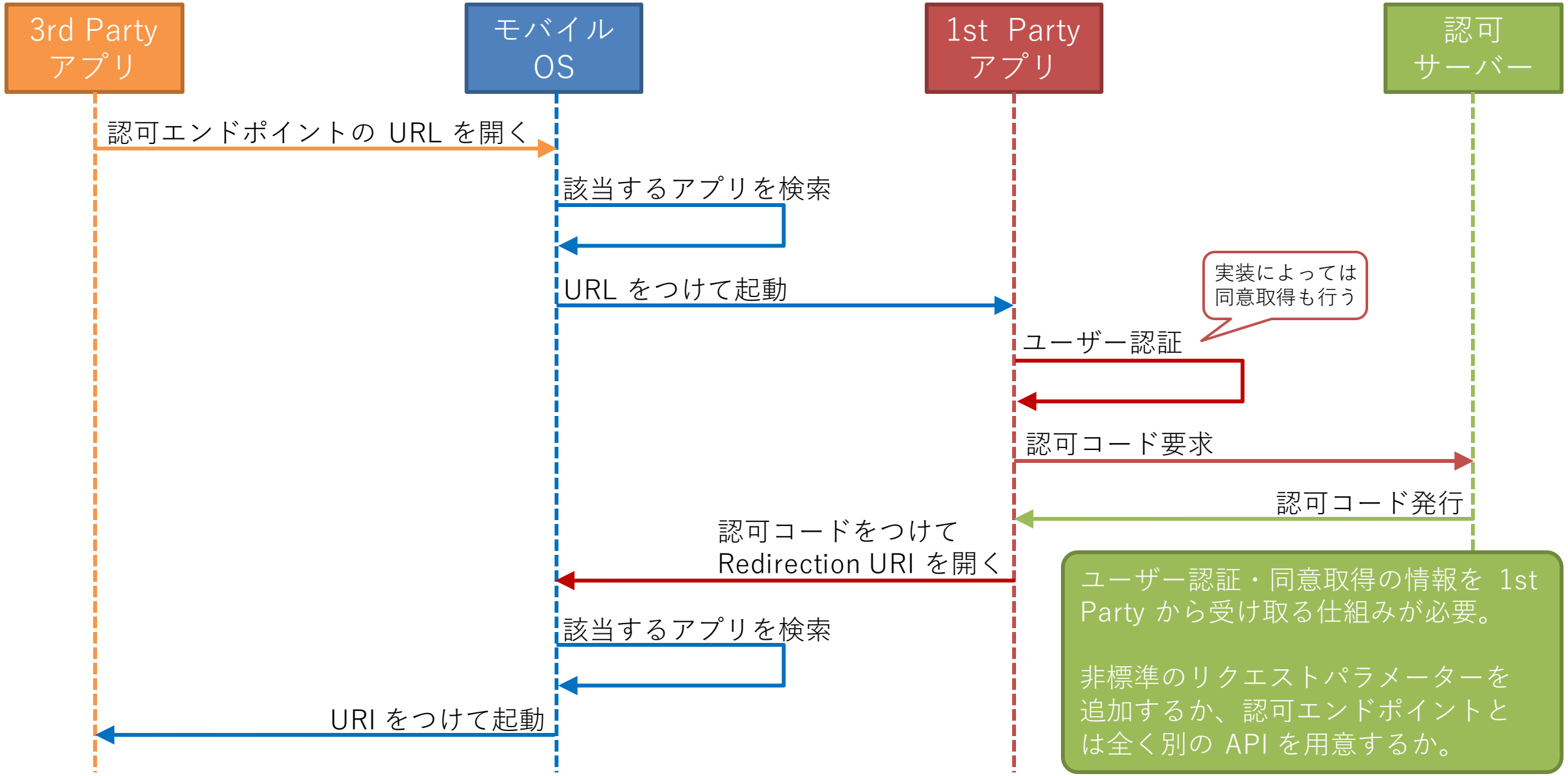
OpenID Certification Conformance Suite 開発責任者。Financial-grade API 仕様策定作業の中心人物の一人。FinTechLabs.io 社 CTO。emobix 社取締役兼 CTO。Authlete 社シニアアーキテクト。組込系出身のスーパーエンジニアで、責任感があり仕事の質も高い上、人柄が大人なので人望も厚い。

- ☑ app2app は、モバイルアプリの OAuth/OIDC フローを改善する実装パターンの一つ。
- ☑ 認可サーバー所有者が提供する First-Party アプリがインストールされている場合に利用可能。
- ☑ 認可リクエストの仲介を、Web ブラウザではなく、First-Party アプリがおこなう。

> 『Claimed "https" Scheme URI Redirection』(BCP 212, Section 7.2) を用い、認可リクエストの URL の処理を First-Party アプリがおこなう。iOS では Deep Linking や Universal Links、Android では App Links として知られている機能。

- ☑ ユーザー認証を First-Party アプリで行う。実装によっては同意取得も First-Party アプリで行う。生体認証等の活用により、UX の改善がはかれる。

app2app


3rd Party
アプリ
モバイル
OS
1st Party
アプリ
認可
サーバー
認可エンドポイントの URL を開く
該当するアプリを検索
URL をつけて起動
実装によっては
同意取得も行う
ユーザー認証
認可コード要求
認可コード発行
認可コードをつけて
Redirection URI を開く
該当するアプリを検索
URI をつけて起動
ユーザー認証・同意取得の情報を 1st Party から受け取る仕組みが必要。
非標準のリクエストパラメーターを追加するか、認可エンドポイントとは全く別の API を用意するか。

Q app2app が急に注目を集めるようになった理由は？

A EBA (European Banking Authority; 欧州銀行監督局) が発行した文書で言及したため。

2019 年 7 月 26 日、EBA は幾つかの問題について解釈方法を明確にするため、文書を発行。

EBA publishes clarifications to the fourth set of issues raised by its Working Group on APIs under PSD2

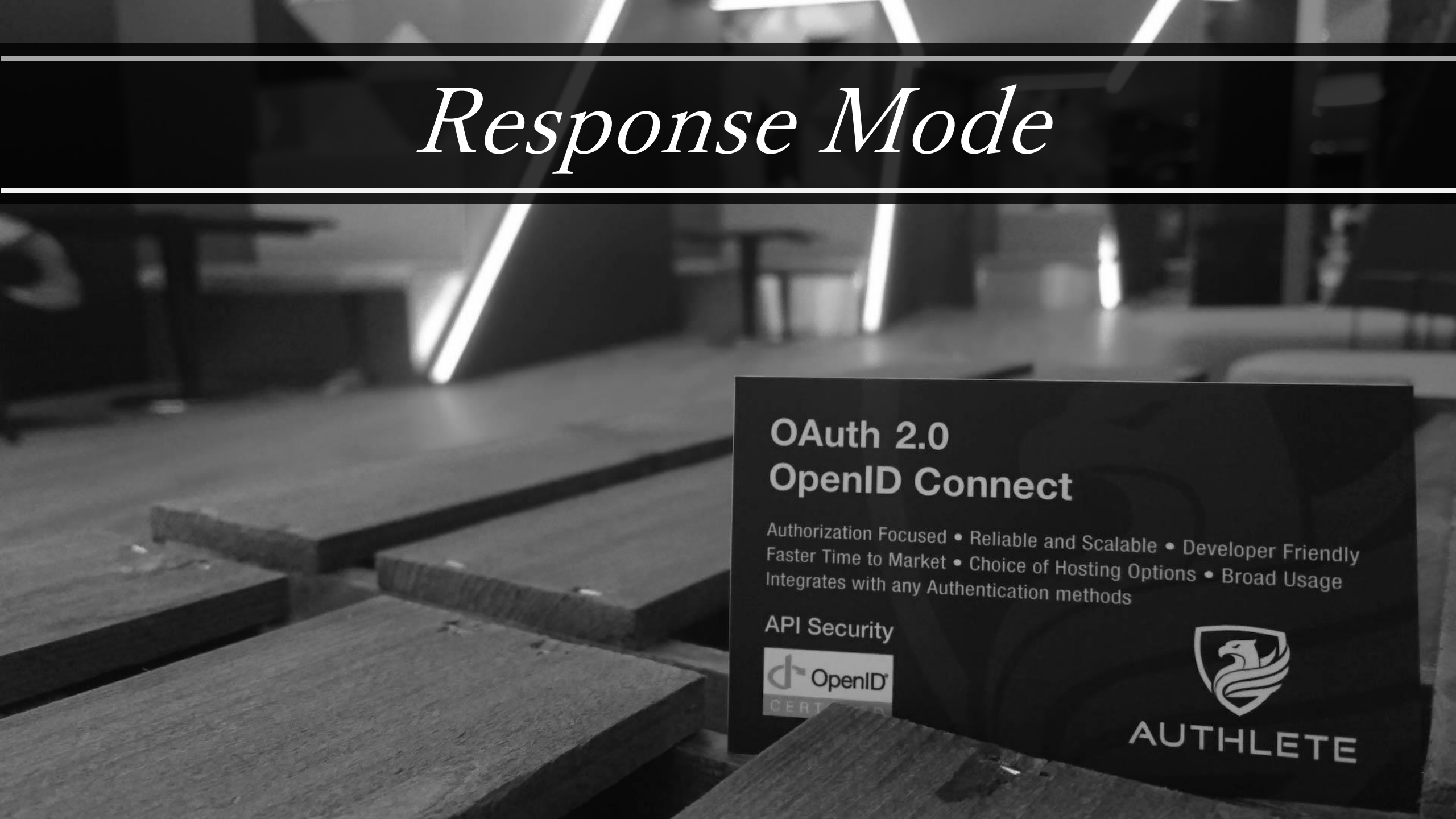
https://eba.europa.eu/eba-publishes-clarifications-to-the-fourth-set-of-issues-raised-by-its-working-group-on-apis-under-psd2

> This means that, ASPSPs that have implemented a redirection approach and that enable their own PSUs to authenticate via the ASPSP's mobile app when the PSU directly accesses his/her account should also support app-to-app redirect when the customer uses a TPP.

ASPSP=Account Servicing Payment Service Provider; PSU=Payment Services User; TPP=Third Party Provider

→ リダイレクションを実装しており、ユーザーが口座に直接アクセスする際に自行モバイルアプリでのユーザー認証を可能としているなら、サードパーティーアプリに対して app-to-app 機能を提供すべきである。

Response Mode



OpenID Connect Core 1.0, Section 3.1.2.1. Authentication Request

response_mode

OPTIONAL. Informs the Authorization Server of the mechanism to be used for returning parameters from the Authorization Endpoint. This use of this parameter is NOT RECOMMENDED when the Response Mode that would be requested is the default mode specified for the Response Type.

→ 認可レスポンスのパラメーター群の返し方を指定するリクエストパラメーター

関連仕様

1	OAuth 2.0 Multiple Response Type Encoding Practices
2	OAuth 2.0 Form Post Response Mode
3	JWT Secured Authorization Response Mode for OAuth 2.0 (JARM)

OAuth 2.0 Multiple Response Type Encoding Practices, 2.1. Response Modes

response_mode の定義値

response_mode=query	パラメーター群をクエリー部に置く
response_mode=fragment	パラメーター群をフラグメント部に置く

response_mode のデフォルト値

response_type	フロー	Response Mode デフォルト値
code	認可コードフロー	query
token	インプリシットフロー	fragment

response_type と response_mode の組み合わせ

response_type	response_mode		
	省略時	query	fragment
none	クエリー	クエリー	フラグメント
code			
token	フラグメント	不可	フラグメント
id_token			
code token			
code id_token			
id_token token			
code id_token token			

OAuth 2.0 Form Post Response Mode

response_mode=form_post	パラメーター群を HTML フォーム内に置く。

```
HTTP/1.1 200 OK  ← 302 Found ではなく 200 OK
Content-Type: text/html;charset=UTF=8  ← HTML を送り返している
Cache-Control: no-cache, no-store
Pragma: no-cache

<html>
 <head><title>Submit This Form</title></head>
 <body onload="javascript:document.forms[0].submit()">  ←JavaScript でフォーム自動送信
  <form method="post" action="https://client.example.org/callback">←Redirection URI
    <input type="hidden" name="state"    value="DcP7csa3hMlvybERqcieLHrRzKBra"/>
    <input type="hidden" name="id_token" value="eyJ（省略）"/>
  </form>                          ↑認可レスポンスパラメーター群
 </body>
</html>
```

OAuth 2.0 Form Post Response Mode, Appendix A より抜粋した認可レスポンスの例（一部調整）

注意点：仕様書の例の通りに実装すると JavaScript が無効になっている場合に詰むので、対策が必要

```
<html>
 <head>
  <meta http-equiv="content-type" content="text/html;charset=UTF-8" />
  <title>認可レスポンス</title>
 </head>
 <body onload="javascript:document.forms[0].submit()">
  <noscript>
   <p>処理を完了させるため、確認ボタンを押してください。</p>
  </noscript>
  <form method="post" action="...">
   <input type="hidden" name="..." value="..." />
   <noscript>
    <input type="submit" value="確認" />
   </noscript>
  </form>
 </body>
</html>
```

JavaScript が無効の場合のみ表示される。ユーザーが手動でボタンを押すことによりフォームが送信される。

response_type と response_mode の組み合わせ（2）

response_type	response_mode			
	省略時	query	fragment	form_post
none	クエリー	クエリー	フラグメント	フォーム
code				
token	フラグメント	不可	フラグメント	フォーム
id_token				
code token				
code id_token				
id_token token				
code id_token token				

JWT Secured Authorization Response Mode for OAuth 2.0 (JARM)

query.jwt	パラメーター群を JWT にまとめ、クエリー部に置く
fragment.jwt	パラメーター群を JWT にまとめ、フラグメント部に置く
form_post.jwt	パラメーター群を JWT にまとめ、HTML フォーム内に置く
jwt	パラメーター群を JWT にまとめ、デフォルトの場所に置く

従来の認可レスポンスの例

```
HTTP/1.1 302 Found
Location: https://example.com/callback?code=123&state=abc
```

JARM の認可レスポンスの形式

```
HTTP/1.1 302 Found
Location: https://example.com/callback?response={JWT}
```

JARM の認可レスポンスの例

```
HTTP/1.1 302 Found
Location: https://client.example.com/cb?response=eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ
9.eyJpc3MiOiJodHRwczovL2FjY291bnRzLmV4YW1wbGUuY29tIiwiYXVkIjoiczZCaGRSa3F0MyIsImV4cC
I6MTMxMTI4MTk3MCwiY29kZSI6IlB5eUZhdXgybzdRMFlmWEJVMzJqaHcuNUZYU1FwdnI4YWt2OUNlUkRTZD
BRQSIsInN0YXRlIjoiUzhOSjd1cWs1Zlk0RWpOdlBfR19GdHlKdTZwVXN2SDlqc1luaTlkTUFKdyJ9.HkdJ_
TYgwBBj10C-aWuNUiA062Amq2b0_oyuc5P0aMTQphAqC2o9WbGSkpfuHVBowlb-zJ15tBvXDIABL_t83q6aj
vjtq_pqsByiRK2dLVdUwKhW3P_9wjvI0K20gdoTNbNlP9Z41mhart4BqraIoI8e-L_EfAHfhCG_DDDv7Yg
```

ペイロード部

```
{
  "iss": "https://accounts.example.com",
  "aud": "s6BhdRkqt3",
  "exp": 1311281970,
  "code": "PyyFaux2o7Q0YfXBU32jhw.5FXSQpvr8akv9CeRDSd0QA",
  "state": "S8NJ7uqk5fY4EjNvP_G_FtyJu6pUsvH9jsYni9dMAJw"
}
```

response_mode=form_post.jwt に対応する認可レスポンス例

```
HTTP/1.1 200 OK
Content-Type: text/html;charset=UTF=8
Cache-Control: no-cache, no-store
Pragma: no-cache

<html>
 <head><title>Submit This Form</title></head>
 <body onload="javascript:document.forms[0].submit()">
  <form method="post" action="https://client.example.com/callback">
    <input type="hidden" name="response"
     value="eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJodHRwczovL2
      FjY291bnRzLmV4YW1wbGUuY29tIiwiYXVkIjoiczZCaGRSa3F0MyIsImV4cCI6MTM
      xMTI4MTk3MCwiYWNjZXNzX3Rva2VuIjoiMllvdG5GWkZFanIxekNzaWNNV3BBQSIs
      InN0YXRlIjoiUzhOSjd1cWs1Zlk0RWpOdlBfR19GdHlKdTZwVXN2SDlqc1luaTlkT
      UFKdyIsInRva2VuX3R5cGUiOiJiZWFyZXIiLCJleHBpcmVzX2luIjoiMzYwMCIsIn
      Njb3BlIjoiZXhhbXBsZSJ9.bgHLOu2dlDjtCnvTLK7hTN_JNwoZXEBnbXQx5vd9z1
      7v1HyzfMqz00Vi002T-SWf2JEs3IVSvAe1xWLIY0TeuaiegklJx_gvB59SQIhXX2i
      fzRmqPoDdmJGaWZ3tnRyFWNnEogJDqGFCo2RHtk8fXkE5IEiBD0g-tN0GS_XnxlE"/>
  </form>
 </body>
</html>
```

Financial-grade API: JWT Secured Authorization Response Mode for OAuth 2.0 (JARM)
4.3.3. Response Mode "form_post.jwt" より抜粋

JARM 関連のクライアントメタデータ (5. Client Metadata)

authorization_signed_response_alg	認可レスポンス JWT の署名アルゴリズム
authorization_encrypted_response_alg	認可レスポンス JWT の暗号アルゴリズム（鍵用）
authorization_encrypted_response_enc	認可レスポンス JWT の暗号アルゴリズム（本文用）

JARM をサポートする認可サーバーのクライアント管理画面には上記メタデータに対応する設定項目がある。

← Authlete の管理コンソールの一部

response_type と response_mode の組み合わせ（3）

response_type	response_mode							
	省略時	query	fragment	form_post	jwt	query.jwt	fragment.jwt	form_post.jwt
Status Code	302	302	302	200	302	302	302	200
JWT	NO	NO	NO	NO	YES	YES	YES	YES
none	クエリー	クエリー	フラグメント	フォーム	クエリー	クエリー	フラグメント	フォーム
code								
token	フラグメント	不可	フラグメント	フォーム	フラグメント	不可	フラグメント	フォーム
id_token								
code token								
code id_token								
id_token token								
code id_token token								

Q JARM 策定前は同等の機能をどう実現していたのか？

A ID トークンを流用。

1. [RP] ID トークンが必要でなくても、ID トークンを要求する。
   具体的には、`response_type` リクエストパラメーターに `id_token` を含める。

2. [AS] 認可レスポンスパラメーター群を用意する。
3. [AS] 認可レスポンスパラメーター群のハッシュ値を計算する。
4. [AS] ハッシュ値群をペイロード部に含む ID トークンを生成する。
5. [AS] 認可レスポンスパラメーター群と ID トークンを返す。

6. [RP] 認可レスポンスパラメーター群のハッシュ値を計算する。
7. [RP] ハッシュ値が ID トークン内のものと一致することを確認する。

認可リクエスト

```
GET /authorize
  ?response_type=code+id_token
  &client_id={クライアントID}
  &redirect_uri={RedirectionURI}
  &scope={スコープ群}
  &state={ステート}
  &nonce={ノンス}
  &code_challenge={チャレンジ}
  &code_challenge_method={メソッド}
  HTTP/1.1
Host: example.com
```

認可レスポンス

```
HTTP/1.1 302 Found
Location: {RedirectionURI}
  #id_token={IDトークン}
  &code={認可コード}
  &state={ステート}
```

ID トークンのペイロード部

```
{
  "iss":     "{発行者}",
  "sub":     "{サブジェクト}",
  "aud":     "{クライアントID}",
  "exp":     有効期間終了日時,
  "iat":     発行日時,
  "nonce":   "{ノンス}",
  "c_hash":  "{認可コードのハッシュ値}",
  "s_hash":  "{ステートのハッシュ値}"
}
```

c_hash	OIDC Core 1.0, Section 3.3.2.11
at_hash	OIDC Core 1.0, Section 3.3.2.11
s_hash	FAPI Part 2, Section 5.1

"*ID Token as a detached signature*"

FAPI Part 2, 5.2.2. Authorization server

2. shall require the `response_type` values `code id_token` or `code id_token token`;
3. shall return ID Token as a detached signature to the authorization response;
4. shall include state hash, `s_hash`, in the ID Token to protect the `state` value if the client supplied a value for `state`. `s_hash` may be omitted from the ID Token returned from the Token Endpoint when `s_hash` is present, in the ID Token returned from the Authorization Endpoint;

→ `response_type` に `id_token` を含むことを要求している。

FAPI Part 2, 5.2.5. JWT Secured Authorization Response Mode

If [JARM] is used to secure the authorization responses, the clauses 2, 3, and 4 of section 5.2.2. do not apply. For example, clients may use [JARM] in conjunction with the response type `code`.

→ JARM を使う場合は `response_type` に `id_token` を含めなくてもよい。

Claims


OAuth 2.0
OpenID Connect
Authorization Focused • Reliable and Scalable • Developer Friendly
Faster Time to Market • Choice of Hosting Options • Broad Usage
Integrates with any Authentication methods
API Security
OpenID
AUTHLETE

OpenID Connect Core 1.0, 5.5. Requesting Claims using the "claims" Request Parameter

claims

OPTIONAL. This parameter is used to request that specific Claims to be returned. The value is a JSON object listing the requested Claims.

→ ID トークンや UserInfo エンドポイントからの応答に含めてほしいクレームを指定する方法

OIDC Core 1.0, Section 5.5 に挙げられている claims リクエストパラメーターの値の例

```
{
 "userinfo":
  {
   "given_name": {"essential": true},
   "nickname": null,
   "email": {"essential": true},
   "email_verified": {"essential": true},
   "picture": null,
   "http://example.info/claims/groups": null
  },
 "id_token":
  {
   "auth_time": {"essential": true },
   "acr": {"values": ["urn:mace:incommon:iap:silver"] }
  }
}
```

ユーザー情報レスポンスに含めてほしいクレーム群

ID トークンに含めてほしいクレーム群

個々のクレーム要求の形式

① `"クレーム名": null` → 制限事項を付けず、単に要求する。

② `"クレーム名": JSON オブジェクト` → 制限事項を付けて要求する。

JSONオブジェクト内に指定する制限事項

`"essential": 真偽値`

→ 必須クレーム（Essential Claim）として要求するかどうかを指定する。

ただし、必須クレームとして要求しても、一部の例外を除き、当該クレームの値を提供できない場合にも OpenID Provider はエラーを返さない。

`"value": クレーム値`

→ 特定の値を返すように指定する。

`"values": [ クレーム値, ... ]`

→ いずれかの値を返すように指定する。クレーム値は希望する順番。

個々のクレーム要求の例

```
"given_name": null
```

→ given_name クレームを要求する。

```
"auth_time": {"essential": true}
```

→ auth_time を必須クレームとして要求する。

```
"sub": {"value": "248289761001"}
```

→ sub クレームの値として 248289761001 を要求する。

sub に対してこのような要求をすると、特定のユーザーの認証を要求することになる。

```
"acr": {"essential": true,
        "values": ["urn:mace:incommon:iap:silver",
                   "urn:mace:incommon:iap:bronze"]}
```

→ acr クレームが "urn:mace:incommon:iap:silver" または "urn:mace:incommon:iap:bronze" のどちらかの値を持つことを要求する。

acr (Authentication Context Class Reference) クレームが必須クレームとして要求された場合は例外的に、指定された値のいずれも満たせなかったときは認証失敗として処理しなければならない。(OIDC Core 1.0, Section 5.5.1.1)

- ✅ クレームによっては多言語対応が可能なものがある。例: `family_name` → Kawasaki, 川崎, カワサキ
- ✅ これらを区別するときは、クレーム名の末尾に『#言語タグ』をつける。

```
"family_name":                  "Kawasaki",
"family_name#ja-Hani-JP":       "川崎",
"family_name#ja-Kana-JP":       "カワサキ",
```

RFC 5646
Tags for Identifying Languages

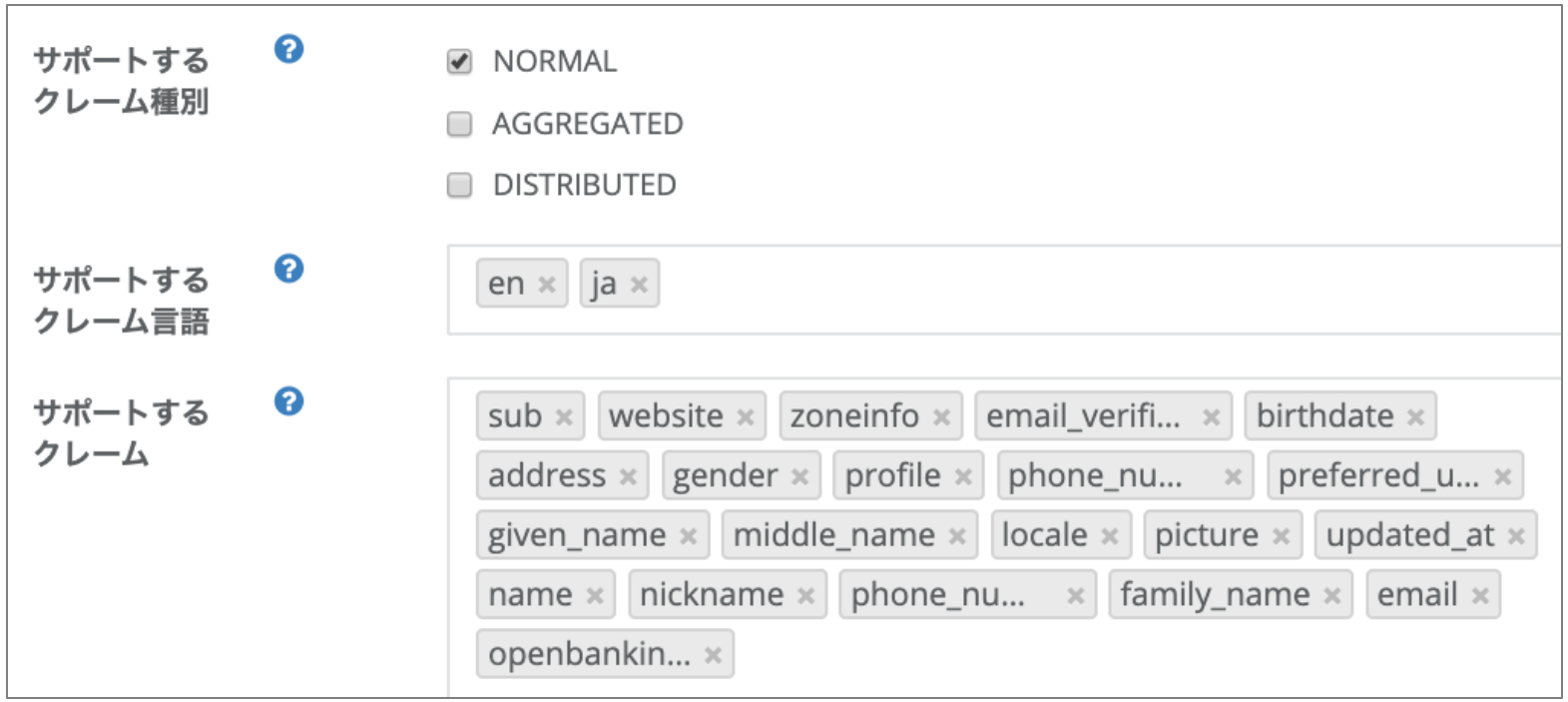
- ✅ 返却されるクレームの言語を指定する場合は、`claims_locales` リクエストパラメーターを使う。

OpenID Connect Core 1.0, Section 5.2. Claims Languages and Scripts

> claims_locales
>
> OPTIONAL. End-User's preferred languages and scripts for Claims being returned, represented as a space-separated list of BCP47 [RFC5646] language tag values, ordered by preference. An error SHOULD NOT result if some or all of the requested locales are not supported by the OpenID Provider.

クレーム関連のサーバーメタデータ

`claims_supported`	サポートするクレーム名のリスト
`claims_locales_supported`	サポートするクレーム用言語タグのリスト
`claims_parameter_supported`	`claims` パラメーターをサポートするか（真偽値）
`claim_types_supported`	サポートするクレーム種別のリスト（OIDC Core 1.0, 5.6. Claim Types）



← Authlete の管理コンソールの一部

User Identification

認可リクエストの対象となるユーザーを指定する方法 (1/2)

1. `claims` リクエストパラメーターを用い、`sub` クレームで `value` を指定する。

```
{
  "id_token": {
    "sub": {"value": "248289761001"}
  }
}
```

2. `login_hint` リクエストパラメーターを用いる。

OpenID Connect Core 1.0, Section 3.1.2.1. Authentication Request

> login_hint
>
> OPTIONAL. Hint to the Authorization Server about the login identifier the End-User might use to log in (if necessary). This hint can be used by an RP if it first asks the End-User for their e-mail address (or other identifier) and then wants to pass that value as a hint to the discovered authorization service. It is RECOMMENDED that the hint value match the value used for discovery. This value MAY also be a phone number in the format specified for the `phone_number` Claim. The use of this parameter is left to the OP's discretion.

認可リクエストの対象となるユーザーを指定する方法 (2/2)

3⃣ `id_token_hint` リクエストパラメーターを用いる。

OpenID Connect Core 1.0, Section 3.1.2.1. Authentication Request

> id_token_hint
>
> OPTIONAL. ID Token previously issued by the Authorization Server being passed as a hint about the End-User's current or past authenticated session with the Client. If the End-User identified by the ID Token is logged in or is logged in by the request, then the Authorization Server returns a positive response; otherwise, it SHOULD return an error, such as `login_required`. When possible, an `id_token_hint` SHOULD be present when `prompt=none` is used and an `invalid_request` error MAY be returned if it is not; however, the server SHOULD respond successfully when possible, even if it is not present. The Authorization Server need not be listed as an audience of the ID Token when it is used as an `id_token_hint` value.
>
> If the ID Token received by the RP from the OP is encrypted, to use it as an `id_token_hint`, the Client MUST decrypt the signed ID Token contained within the encrypted ID Token. The Client MAY re-encrypt the signed ID token to the Authentication Server using a key that enables the server to decrypt the ID Token, and use the re-encrypted ID token as the `id_token_hint` value.

User Authentication

認証後許容経過時間（max age）

ユーザーが既にログイン済みでも、最後のユーザー認証処理から一定の時間が経過していた場合に再認証を要求する方法が提供されている。

OpenID Connect Core 1.0, Section 3.1.2.1. Authentication Request

> max_age
>
> OPTIONAL. Maximum Authentication Age. Specifies the allowable elapsed time in seconds since the last time the End-User was actively authenticated by the OP. If the elapsed time is greater than this value, the OP MUST attempt to actively re-authenticate the End-User. (The `max_age` request parameter corresponds to the OpenID 2.0 PAPE [OpenID.PAPE] `max_auth_age` request parameter.) When `max_age` is used, the ID Token returned MUST include an `auth_time` Claim Value.

関連するクライアントメタデータ

`default_max_age`	`max_age` が省略された場合のデフォルト値

再認証・同意確認制御（prompt）

OpenID Connect Core 1.0, Section 3.1.2.1. Authentication Request

> prompt
>
> OPTIONAL. Space delimited, case sensitive list of ASCII string values that specifies whether the Authorization Server prompts the End-User for reauthentication and consent. The defined values are:

`prompt` リクエストパラメーター用に定義済みの値

`login`	既にログイン済みかどうかに関わらず、ユーザー認証を改めて要求する。
`consent`	既に同意済みかどうかに関わらず、同意確認を改めて要求する。
`select_account`	複数のアカウントの中から選択させる。
`none`	ユーザー認証や同意取得用の UI を表示してはならない。`prompt=none` を指定すると、ユーザーが既にログイン済みで、かつ、要求している権限に対して同意済みでない限り、認可リクエストは失敗する。`prompt` に `none` が含まれている場合、他の値は含まれていてはならない。

⚠ `create` という値も提案されている。ログイン UI のかわりにアカウント作成 UI の表示を促す。
`https://bitbucket.org/openid/connect/src/master/openid-connect-prompt-create-1_0.xml`

認証コンテキスト（authentication context）

ユーザー認証で満たしてほしい認証コンテキストを指定する方法が定義されている。
希望する認証コンテキストを `acr` クレームの値として要求すればよい。

OpenID Connect Core 1.0, Section 2. ID Token

> acr
>
> OPTIONAL. Authentication Context Class Reference. String specifying an Authentication Context Class Reference value that identifies the Authentication Context Class that the authentication performed satisfied. The value "0" indicates the End-User authentication did not meet the requirements of ISO/IEC 29115 [ISO29115] level 1. Authentication using a long-lived browser cookie, for instance, is one example where the use of "level 0" is appropriate. Authentications with level 0 SHOULD NOT be used to authorize access to any resource of any monetary value. (This corresponds to the OpenID 2.0 PAPE [OpenID.PAPE] `nist_auth_level` 0.) An absolute URI or an RFC 6711 [RFC6711] registered name SHOULD be used as the `acr` value; registered names MUST NOT be used with a different meaning than that which is registered. Parties using this claim will need to agree upon the meanings of the values used, which may be context-specific. The acr value is a case sensitive string.
>
> 注：認可リクエストパラメーターではなく、IDトークン内のクレーム

→ ユーザー認証が満たした認証コンテキストクラス。ユーザー認証の強度に関わる。

☑ claims リクエストパラメーターを用いて acr クレームを要求する方法は説明済み。

```
"acr": {"essential": true,
        "values": ["urn:mace:incommon:iap:silver",
                   "urn:mace:incommon:iap:bronze"]}
```

☑ acr クレームを任意クレーム（Voluntary Claim）として要求するなら、簡易的な方法として acr_values リクエストパラメーターを利用できる。

OpenID Connect Core 1.0, Section 3.1.2.1. Authentication Request

> **acr_values**
>
> OPTIONAL. Requested Authentication Context Class Reference values. Space-separated string that specifies the acr values that the Authorization Server is being requested to use for processing this Authentication Request, with the values appearing in order of preference. The Authentication Context Class satisfied by the authentication performed is returned as the acr Claim Value, as specified in Section 2. The acr Claim is requested as a Voluntary Claim by this parameter.

→ acr クレームの値をスペース区切りで並べる。

FAPI Part 2, 5.2.3. Public client

3. shall require user authentication at LoA 3 or greater by requesting the `acr` claim as an essential claim as defined in section 5.5.1.1 of OIDC;

→ FAPI Part 2 では `acr` を必須クレームとして要求しなければならない。

しかし、FAPI Part 2 準拠のはずの英国オープンバンキングプロファイル仕様では、この規定は適用されない。

英国オープンバンキング → `https://www.openbanking.org.uk`

英国オープンバンキングプロファイル（OBP）仕様策定時、FAPI もまた同時並行で仕様策定作業が進められていた。OBP 策定時に参照した FAPI 仕様のスナップショットに当該規定がまだ含まれていなかったことが原因だと思われる。

OpenID Certification Conformance Suite 開発を請け負っている FinTechLabs.io 社は、当時、同スイートに英国オープンバンキングプロファイル Certification 用のテストケースを追加する作業をおこなっていた。

テストスイート自体のテストは（現在も）Authlete を使っておこなっており・・・

joseph

acr を必須クレームとして要求していない、との理由で、オープンバンキングのテストケースが失敗してしまう。

taka

FAPI Part 2 にそのように書かれているからね。テストケースのほうを修正すべきだね。

joseph

それが、オープンバンキングプロファイルの仕様にはその規定が無いんだ。

taka

FAPI Part 2 準拠を謳うのであれば、OBP の仕様を修正すべきだね。

joseph

時期的に仕様変更は難しくて…。acr を必須クレームとして要求することを求めると、全銀行の OBP 実装がことごとく Certification テストをパスできなくなると思う。

止むを得ないね。OBP 用のフラグを用意し、そのフラグが立っているときは、acr が必須クレームとして要求されていることを確認しないようにするよ。

taka

この後、Authlete のサーバー管理コンソールには OPEN_BANKING フラグが追加された。

副作用で、Authlete は英国オープンバンキングのセキュリティプロファイル準拠製品としてリストされることになったよ。
https://openbanking.atlassian.net/wiki/spaces/DZ/pages/126321042/Open+Banking+Security+Profile+Conformance

ACR 関連のクライアントメタデータ

`default_acr_values`	認可リクエスト用のデフォルトの ACR クレーム値リスト

↑ Authlete のクライアント管理コンソールの一部

`default_acr_values` をセットしておけば、`claims` や `acr_values` を用いなくても、`acr` クレームを要求することは可能。ただし、柔軟性には欠ける。

ACR 関連のサーバーメタデータ

`acr_values_supported`	サポートする ACR クレーム値のリスト

↑ Authlete のサーバー管理コンソールの一部

User Interface

ユーザーインターフェース言語 (UI locales)

ユーザー認証画面や同意確認画面等の UI で使用する言語を指定する方法が提供されている。

OpenID Connect Core 1.0, Section 3.1.2.1. Authentication Request

ui_locales
OPTIONAL. End-User's preferred languages and scripts for the user interface, represented as a space-separated list of BCP47 [RFC5646] language tag values, ordered by preference. For instance, the value "fr-CA fr en" represents a preference for French as spoken in Canada, then French (without a region designation), followed by English (without a region designation). An error SHOULD NOT result if some or all of the requested locales are not supported by the OpenID Provider.

関連するサーバーメタデータ

`ui_locales_supported`	サポートするユーザーインターフェース用言語のリスト

←Authlete のサーバー管理コンソールの一部

ディスプレイ（display）

表示装置のタイプを指定する方法が提供されている。

OpenID Connect Core 1.0, Section 3.1.2.1. Authentication Request

> display
>
> OPTIONAL. ASCII string value that specifies how the Authorization Server displays the authentication and consent user interface pages to the End-User. The defined values are:

`display` リクエストパラメーター用に定義済みの値

`page`	ユーザーエージェントのページ全体表示向け
`popup`	ユーザーエージェントのポップアップ表示向け
`touch`	タッチデバイス向け
`wap`	フィーチャーフォン向け

↑ Authlete の管理コンソールの一部

関連するサーバーメタデータ

`display_types_supported`	サポートする表示装置タイプのリスト

Protection


OAuth 2.0
OpenID Connect
Authorization Focused • Reliable and Scalable • Developer Friendly
Faster Time to Market • Choice of Hosting Options • Broad Usage
Integrates with any Authentication methods
API Security
OpenID
AUTHLETE

`code_challenge`, `code_challenge_method` (RFC 7636 : PKCE)

動画	**OAuth & OIDC 勉強会【入門編】**1:15:05～ https://www.youtube.com/watch?v=PKPj_MmLq5E&t=4505 https://www.authlete.com/ja/resources/videos/20200317　（資料ダウンロード）
記事	**PKCE: 認可コード横取り攻撃対策のために OAuth サーバーとクライアントが実装すべきこと** https://qiita.com/TakahikoKawasaki/items/00f333c72ed96c4da659

`state` (RFC 6749 : The OAuth 2.0 Authorization Framework)

- ☑ CSRF 攻撃を防ぐために用いる。(RFC 6819, 3.6, 4.4.1.8, 5.3.5)
- ☑ セッションに紐付く値を指定する。セッションに無関係なランダム値を指定してはダメ。
- ☑ 認可リクエストの `state` パラメーターの値がそのまま認可レスポンスに含まれて返ってくるので、その値がセッションと紐付いていることを確認する。
- ☑ FAPI Part 2 で、認可レスポンスに ID トークンが含まれる場合、`state` のハッシュ値を表す `s_hash` クレームが ID トークン内に含まれる。JARM を使っていない場合、`state` レスポンスパラメーターのハッシュ値を計算し、`s_hash` の値と一致することを確認する。

nonce (OpenID Connect Core 1.0, Section 3.1.2.1)

- ☑ セッションと ID トークンの紐付け、およびレプレイ攻撃対策のために用いる。
- ☑ 認可リクエストの `nonce` パラメーターの値がそのまま ID トークンに含まれて返ってくるので、その値が認可リクエストに含めた値と一致することを確認する。

PKCE vs. Nonce: Equivalent or Not?

https://danielfett.de/2020/05/16/pkce-vs-nonce-equivalent-or-not/

Dr. Daniel Fett @dfett42

yes.com 社の Web セキュリティ研究者、セキュリティスペシャリスト。
Web プロトコルの形式的分析の研究で Stuttgart 大学で博士号取得。

Others


OAuth 2.0
OpenID Connect
Authorization Focused • Reliable and Scalable • Developer Friendly
Faster Time to Market • Choice of Hosting Options • Broad Usage
Integrates with any Authentication methods
API Security
OpenID
AUTHLETE

`authorization_details` (OAuth 2.0 Rich Authorization Requests : RAR)

動画	**OAuth & OIDC 勉強会【アクセストークン編】#6** 5:57～ https://www.youtube.com/watch?v=g0nzidOMFGk&t=357 https://www.authlete.com/ja/resources/videos/20200422/　(資料ダウンロード)

`resource` (RFC 8707 : Resource Indicators for OAuth 2.0)

動画	**OAuth & OIDC 勉強会【アクセストークン編】#6** 0:04～ https://www.youtube.com/watch?v=g0nzidOMFGk&t=4 https://www.authlete.com/ja/resources/videos/20200422/　(資料ダウンロード)
記事	**Resource Indicators for OAuth 2.0** https://qiita.com/TakahikoKawasaki/items/57d6242ff8b20dbff08f

Thank You

お問い合わせ

https://www.authlete.com/ja/contact/

一般	info@authlete.com
営業	sales@authlete.com
広報	pr@authlete.com
技術	support@authlete.com

@authlete_jp