「挫折しないOAuth/OpenID Connect入門」の理解を深める会

Webinar on October 6, 2021

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

Software Design 2021 年 10 月号

https://gihyo.jp/magazine/SD/archive/2021/202110

第 2 特集

挫折しないOAuth/OpenID Connect入門

APIを守る認証・認可フローのしくみ

第 1 章 OAuthとは、OpenID Connectとは
- OAuthとは
- OpenID Connectとは
- OAuth 2.0とOpenID Connectの関係

第 2 章 知っておきたい仕様と規格
- 認可コードフロー
- 認可リクエストと認可レスポンス
- トークンリクエストとトークンレスポンス
- クライアント認証
- IDトークン、JWT、JWS／JWE

第 3 章 トークンハンドリングの基本
- アクセストークンの使い方
- リフレッシュトークンを利用したアクセストークンの再取得
- ユーザー情報エンドポイント
- IDトークンの署名検証
- アクセストークンの検証
- アクセストークンの失効

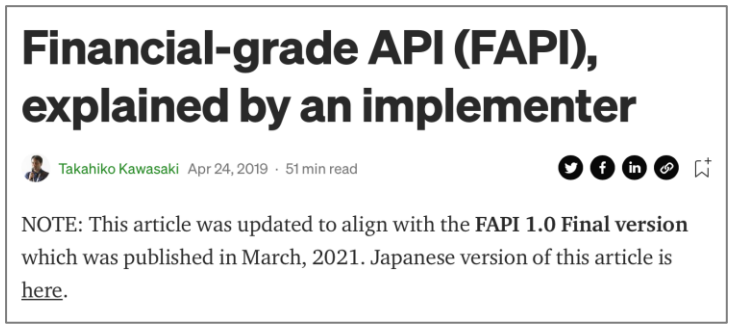
著者紹介：川崎貴彦

Authlete社代表取締役社長。ソフトウェアエンジニア。OAuth/OpenID Connect関連仕様の策定作業に参画し、実装に落とし込む仕事をしている。

名前が掲載されている最近の仕様

- Financial-grade API Security Profile (FAPI 1.0)
- Financial-grade API 2.0 Baseline Profile (FAPI 2.0 Baseline)
- Client-Initiated Backchannel Authentication (CIBA)
- OpenID Connect for Identity Assurance (OIDC4IDA)
- Grant Management for OAuth 2.0 (GM)
- RFC 8705 OAuth 2.0 Mutual-TLS Client Authentication and Certificate-Bound Access Tokens (MTLS)
- RFC 9126 OAuth 2.0 Pushed Authorization Requests (PAR)
- OAuth 2.0 Rich Authorization Requests (RAR)

日英両方で技術ブログを執筆。Financial-grade API解説記事の英語版はOpen Foundation公式ウェブサイトに転載され、ブラジルオープンバンキング用にポルトガル語にも翻訳されている。『一番分かりやすいOAuthの説明』は2017年に公開されたQiita記事中の人気一位。



FAPI解説記事(英語原文) ※1

FAPI解説記事(ポルトガル語翻訳) ※2

https://qiita.com/TakahikoKawasaki/items/e37caf50776e00e733be

※1: https://darutk.medium.com/financial-grade-api-fapi-explained-by-an-implementer-d09fcf2ff932
※2: https://openid.net/financial-grade-api-fapi-explicada-por-um-desenvolvedor/

企業紹介：Authlete（オースリート）

2015年9月創業。OAuth/OpenID Connect関連仕様のサーバ側実装を提供。最新仕様の実装速度・実装品質が強み。特にFinancial-grade API (FAPI) セキュリティプロファイルの実装としては世界トップクラスで、国内外に金融機関の顧客を抱える。

金融業界の顧客の例

- Nubank (世界最大のデジタル専業銀行)
- BTG Pactual (ラテンアメリカ最大の投資銀行)
- みんなの銀行 (日本発デジタル専業銀行)
- セブン銀行
- 楽天銀行
- LINE Bank設立準備株式会社
- SBIデジトラスト
- トヨタファイナンス

Joseph Heenan

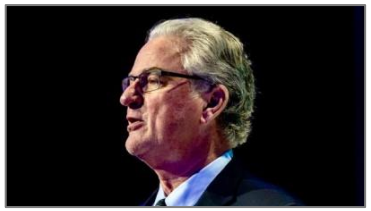
Lead of the OpenID Certification Team. Active contributor of the FAPI WG.

Justin Richer

Author of "OAuth 2 in Action" and many OAuth/OIDC-related specifications.

Nat Sakimura

Chairman of the OpenID Foundation.



Don Thibeau

Vice Chair of the Open Identity Exchange.

第1章 OAuthとは、OpenID Connectとは

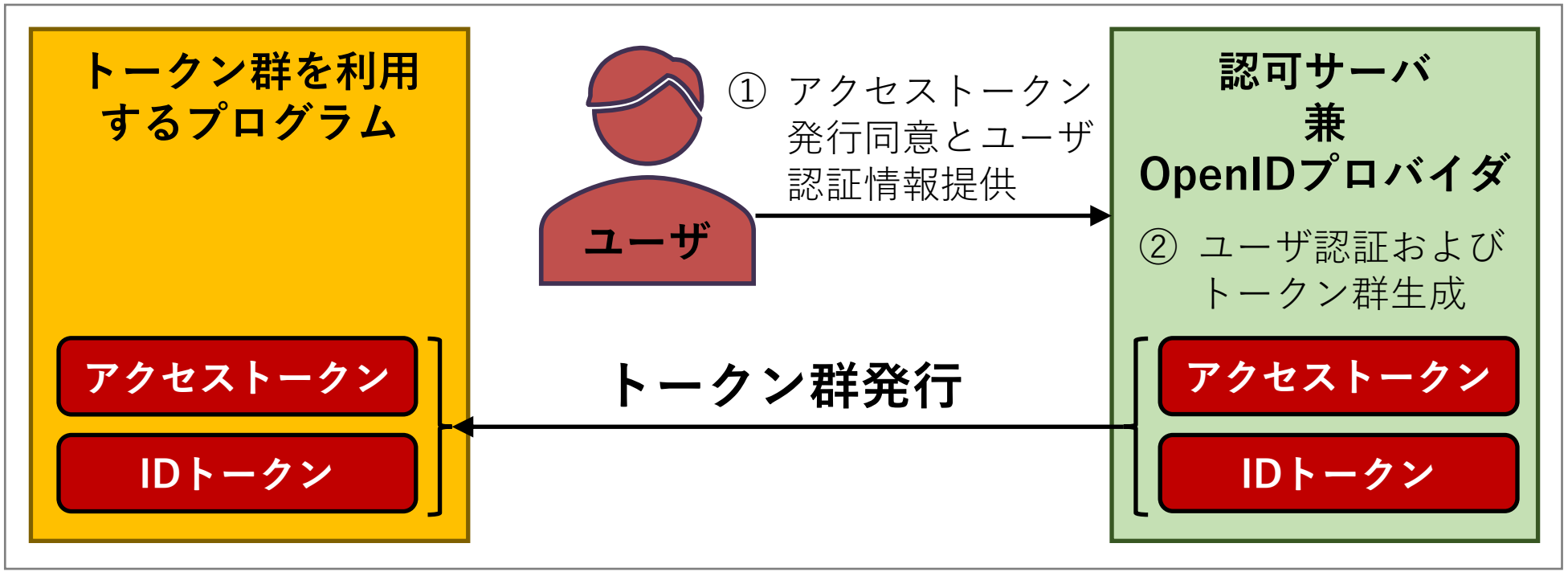
超要約

OAuth 2.0 → **アクセストークン**を発行する手順を定めた技術仕様

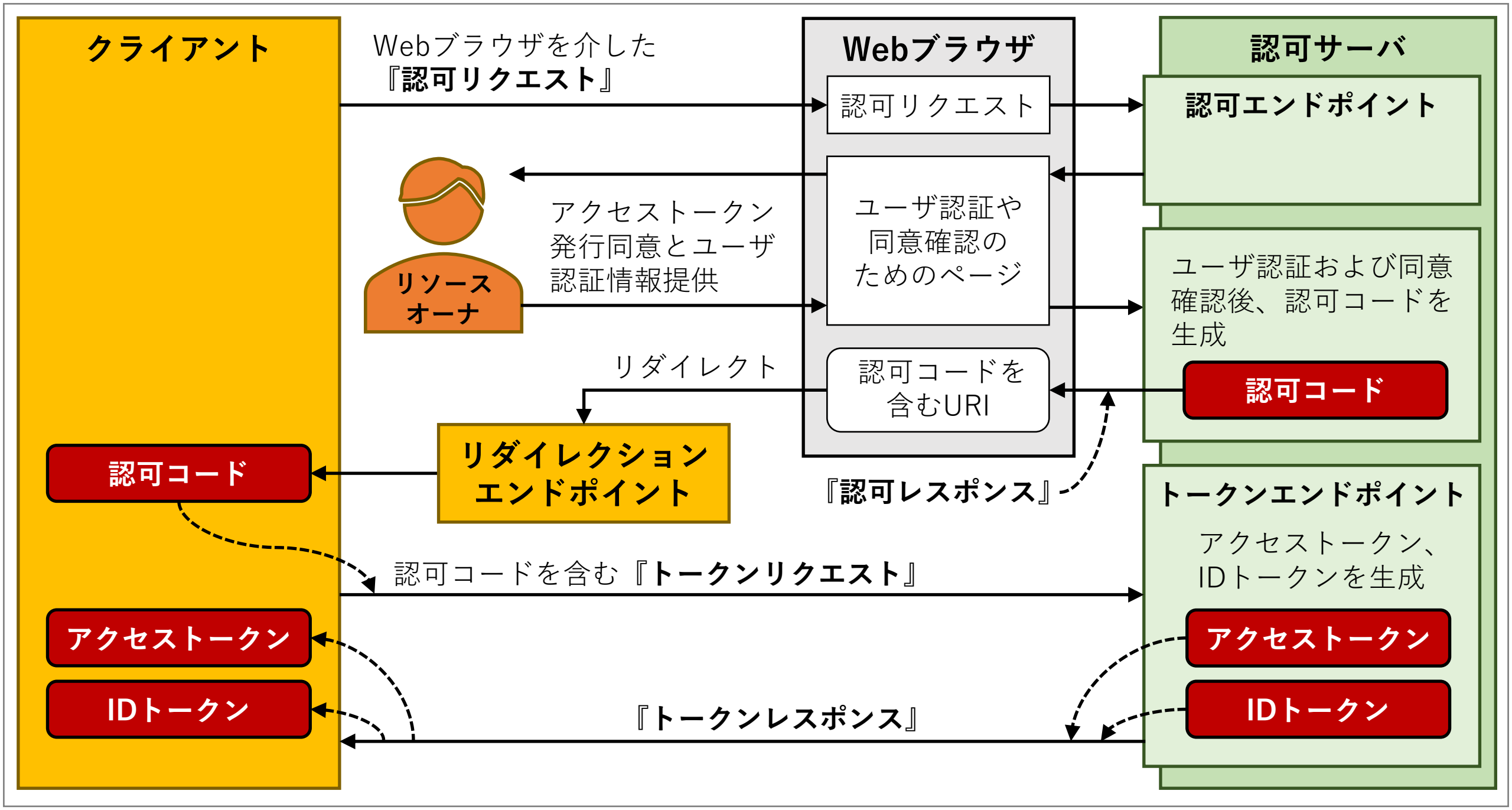
- Web API を呼び出すときに使う。

OpenID Connect → **IDトークン**を発行する手順を定めた技術仕様

- ユーザー認証結果やユーザー属性情報を利用したいときに使う。シングルサインオンなど。



第2章 知っておきたい仕様と規格

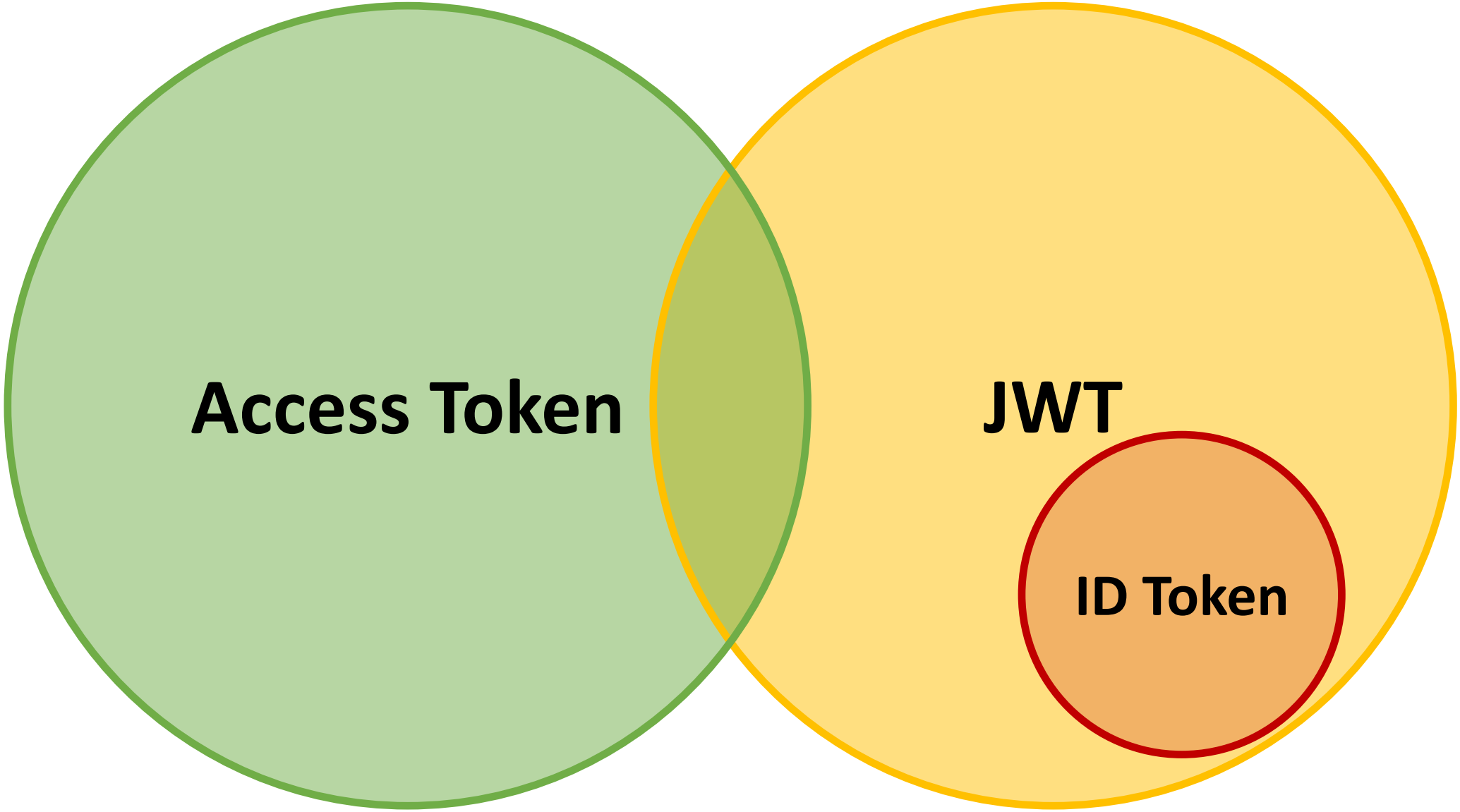


IDトークン発行を伴う認可コードフロー

トークンレスポンスの例

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

{
  "access_token":  "i_1X-euOC6-45zdObsQty7hgDMW9RUTjSGb0pzP69X0",
  "refresh_token": "RnxcnFmOufZBpyqRGOtgf5LEzYvyPVnUgw7yAeGsSTA",
  "scope": "email openid",
  "id_token": "eyJraWQiOiJTRC0yMDIxLTEwIiwiYWxnIjoiRVMyNTYifQ.
              eyJlbWFpbCI6ImpvaG5AZXhhbXBsZS5jb20iLCJpc3MiOi
              JodHRwczovL2V4YW1wbGUuY29tIiwic3ViIjoiMTAwMSIs
              ImF1ZCI6WyIyMTM0ODg2NzI4MjYiXSwiZXhwIjoxNjI4Mz
              k4NTEwLCJpYXQiOjE2MjgzOTQ5MTB9.
              46wrzNIBOpgpn6sDKDvZQ9qgP9hcdPDyHrP7pGjQ7hga0P
              oCqb_dGyS5K-kRgN6RmZwcXYDCa-eg3t8TSHMCkQ",
  "token_type": "Bearer",
  "expires_in": 86400
}
```


Access Token
JWT
ID Token

```
eyJraWQiOiIxZTlnZGs3IiwiYWxnIjoiUlMyNTYifQ.ewogImlz
cyI6ICJodHRwOi8vc2VydmVyLmV4YW1wbGUuY29tIiwKICJzdWIiOiAiMjQ4
Mjg5NzYxMDAxIiwKICJhdWQiOiAiczZCaGRSa3F0MyIsCiAibm9uY2UiOiAi
bi0wUzZfV3pBMk1qIiwKICJleHAiOiAxMzExMjgxOTcwLAogImlhdCI6IDEz
MTEyODA5NzAsCiAibmFtZSI6ICJKYW5lIERvZSIsCiAiZ2l2ZW5fbmFtZSI6
ICJKYW5lIiwKICJmYW1pbHlfbmFtZSI6ICJEb2UiLAogImdlbmRlciI6ICJm
ZW1hbGUiLAogImJpcnRoZGF0ZSI6ICIwMDAwLTEwLTMxIiwKICJlbWFpbCI6
ICJqYW5lZG9lQGV4YW1wbGUuY29tIiwKICJwaWN0dXJlIjogImh0dHA6Ly9l
eGFtcGxlLmNvbS9qYW5lZG9lL21lLmpwZyIKfQ.rHQjEmBqn9Jre0OLykYNn
spA10Qql2rvx4FsD00jwlB0Sym4NzpgvPKsDjn_wMkHxcp6CilPcoKrWHcip
R2iAjzLvDNAReF97zoJqq880ZD1bwY82JDauCXELVR9O6_B0w3K-E7yM2mac
AAgNCUwtik6SjoSUZRcf-O5lygIyLENx882p6MtmwaL1hd6qn5RZOQ0TLrOY
u0532g9Exxcm-ChymrB4xLykpDj3lUivJt63eEGGN6DH5K6o33TcxkIjNrCD
4XB1CKKumZvCedgHHF3IAK4dVEDSUoGlH9z4pP_eWYNXvqQOjGs-rDaQzUHl
6cQQWNiDpWOl_lxXjQEvQ
```

(*“OpenID Connect Core 1.0, A.2. Example using response_type=id_token” より抜粋*)

ヘッダー . ペイロード . 署名

“RFC 7515 (JSON Web Signature (JWS)), 7.1. JWS Compact Serialization”

```
BASE64URL(UTF8(JWS Protected Header)) || '.' ||
BASE64URL(JWS Payload) || '.' ||
BASE64URL(JWS Signature)
```

ヘッダー
ペイロード
署名

各部分は base64url で
エンコードされている。

JWS形式の
ID トークン

base64url でデコード

base64url でデコード

```
eyJraWQiOiIxZTlnZGs3IiwiYWxnIjoiUlMyNTYifQ.ewogImlz
cyI6ICJodHRwOi8vc2VydmVyLmV4YW1wbGUuY29tIiwKICJzdWIiOiAiMjQ4
Mjg5NzYxMDAxIiwKICJhdWQiOiAiczZCaGRSa3F0MyIsCiAibm9uY2UiOiAi
bi0wUzZfV3pBMk1qIiwKICJleHAiOiAxMzExMjgxOTcwLAogImlhdCI6IDEz
MTEyODA5NzAsCiAibmFtZSI6ICJKYW5lIERvZSIsCiAiZ2l2ZW5fbmFtZSI6
ICJKYW5lIiwKICJmYW1pbHlfbmFtZSI6ICJEb2UiLAogImdlbmRlciI6ICJm
ZW1hbGUiLAogImJpcnRoZGF0ZSI6ICIwMDAwLTEwLTMxIiwKICJlbWFpbCI6
ICJqYW5lZG9lQGV4YW1wbGUuY29tIiwKICJwaWN0dXJlIjogImh0dHA6Ly9l
eGFtcGxlLmNvbS9qYW5lZG9lL21lLmpwZyIKfQ.rHQjEmBqn9Jre0OLykYNn
spA10Qql2rvx4FsD00jwlB0Sym4NzpgvPKsDjn_wMkHxcp6CilPcoKrWHcip
R2iAjzLvDNAReF97zoJqq880ZD1bwY82JDauCXELVR9O6_B0w3K-E7yM2mac
AAgNCUwtik6SjoSUZRcf-O5lygIyLENx882p6MtmwaL1hd6qn5RZOQ0TLrOY
u0532g9Exxcm-ChymrB4xLykpDj3lUivJt63eEGGN6DH5K6o33TcxkIjNrCD
4XB1CKKumZvCedgHHF3IAK4dVEDSUoGlH9z4pP_eWYNXvqQOjGs-rDaQzUHl
6cQQWNiDpWOl_lxXjQEvQ
```

"OpenID Connect Core 1.0, A.2. Example using response_type=id_token"

base64url でデコード

```
{"kid":"1e9gdk7","alg":"RS256"}
```

```
{
 "iss": "http://server.example.com",
 "sub": "248289761001",
 "aud": "s6BhdRkqt3",
 "nonce": "n-0S6_WzA2Mj",
 "exp": 1311281970,
 "iat": 1311280970,
 "name": "Jane Doe",
 "given_name": "Jane",
 "family_name": "Doe",
 "gender": "female",
 "birthdate": "0000-10-31",
 "email": "janedoe@example.com",
 "picture": "http://example.com/janedoe/me.jpg"
}
```

(バイナリデータ)

ヘッダー `{"kid":"1e9gdk7","alg":"RS256"}`

alg	Algorithm
HS256	HMAC using SHA-256
HS384	HMAC using SHA-384
HS512	HMAC using SHA-512
RS256	RSASSA-PKCS1-v1_5 using SHA-256
RS384	RSASSA-PKCS1-v1_5 using SHA-384
RS512	RSASSA-PKCS1-v1_5 using SHA-512
ES256	ECDSA using P-256 and SHA-256
ES384	ECDSA using P-384 and SHA-384
ES512	ECDSA using P-521 and SHA-512
PS256	RSASSA-PSS using SHA-256 and MGF1 with SHA-256
PS384	RSASSA-PSS using SHA-384 and MGF1 with SHA-384
PS512	RSASSA-PSS using SHA-512 and MGF1 with SHA-512
none	No digital signature or MAC performed

JWS のヘッダー部分は JSON です。

alg は 署名アルゴリズムを表す必須のパラメーター。

alg を含む全てのパラメーターは RFC 7515 (JWS) に列挙されています。

```
1.alg   7.x5t
2.jku   8.x5t#S256
3.jwk   9.typ
4.kid   10.cty
5.x5u   11.crit
6.x5c
```

RFC 7518 (JSON Web Algorithms (JWA)), 3.1. “alg” (Algorithm) Header Parameter Values for JWS

ペイロード（JWS）

任意のバイト列

ペイロード（IDトークン）

```
{
 "iss": "http://server.example.com",
 "sub": "248289761001",
 "aud": "s6BhdRkqt3",
 "nonce": "n-0S6_WzA2Mj",
 "exp": 1311281970,
 "iat": 1311280970,
 "name": "Jane Doe",
 "given_name": "Jane",
 "family_name": "Doe",
 "gender": "female",
 "birthdate": "0000-10-31",
 "email": "janedoe@example.com",
 "picture": "http://example.com/janedoe/me.jpg"
}
```

JWS

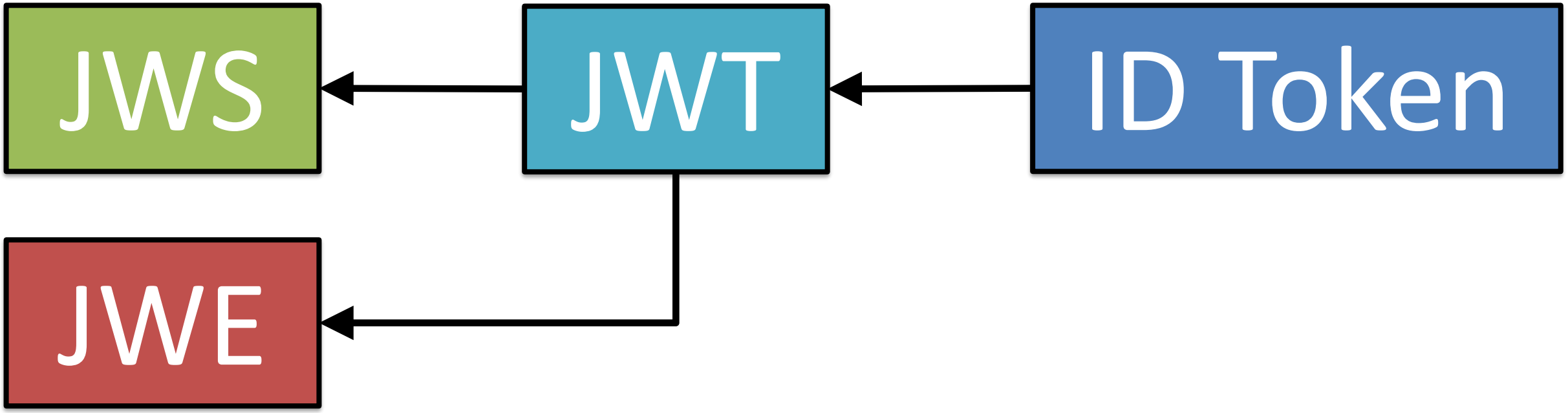
RFC 7515 (JWS) 自体はペイロードが JSON であることを要求しません。

JWT

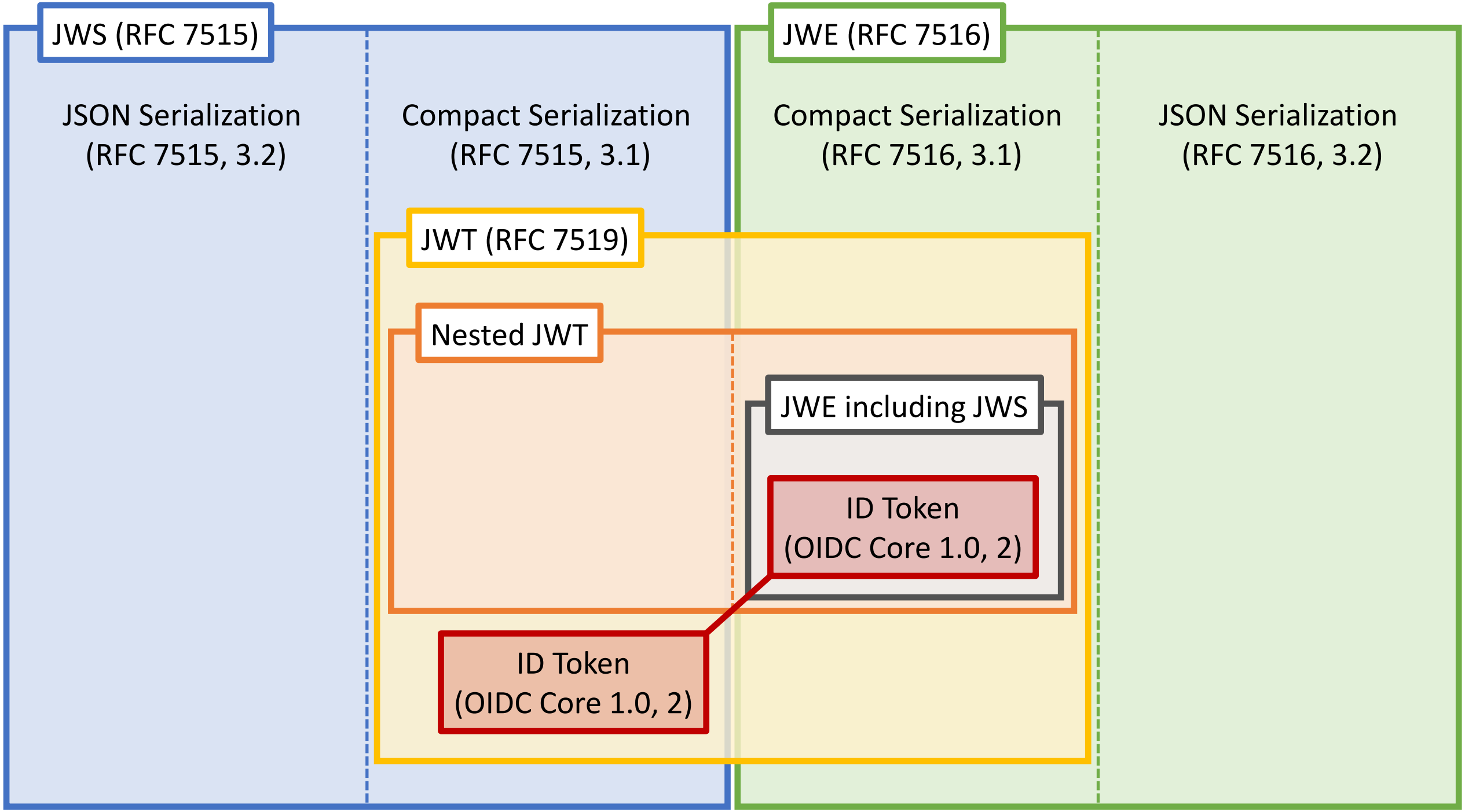
JSON を要求し、いくつかのパラメーターを定義しているのは、RFC 7519 (JSON Web Token (JWT)) です。

ID Token

OpenID Connect は JWT を拡張し、IDトークンを定義しました。



JWS	JSON Web Signature	署名
JWT	JSON Web Token	トークン
JWE	JSON Web Encryption	暗号


JWS (RFC 7515)
JWE (RFC 7516)
JSON Serialization
(RFC 7515, 3.2)
Compact Serialization
(RFC 7515, 3.1)
Compact Serialization
(RFC 7516, 3.1)
JSON Serialization
(RFC 7516, 3.2)
JWT (RFC 7519)
Nested JWT
JWE including JWS
ID Token
(OIDC Core 1.0, 2)
ID Token
(OIDC Core 1.0, 2)

第3章 トークンハンドリングの基本

アクセストークンの使い方

RFC 6750 (The OAuth 2.0 Authorization Framework: Bearer Token Usage) には、Protected Resource Endpoint（いわゆる Web API）へのリクエストにアクセストークンを含める方法が三つ定義されている。

① **Authorization Request Header Field** (Section 2.1)

→ HTTP ヘッダー `Authorization` にアクセストークンを埋め込む

② **Form-Encoded Body Parameter** (Section 2.2)

→ フォームパラメーター `access_token` を使う

③ **URI Query Parameter** (Section 2.3)

→ クエリーパラメーター `access_token` を使う

① Authorization Request Header Field (Section 2.1)

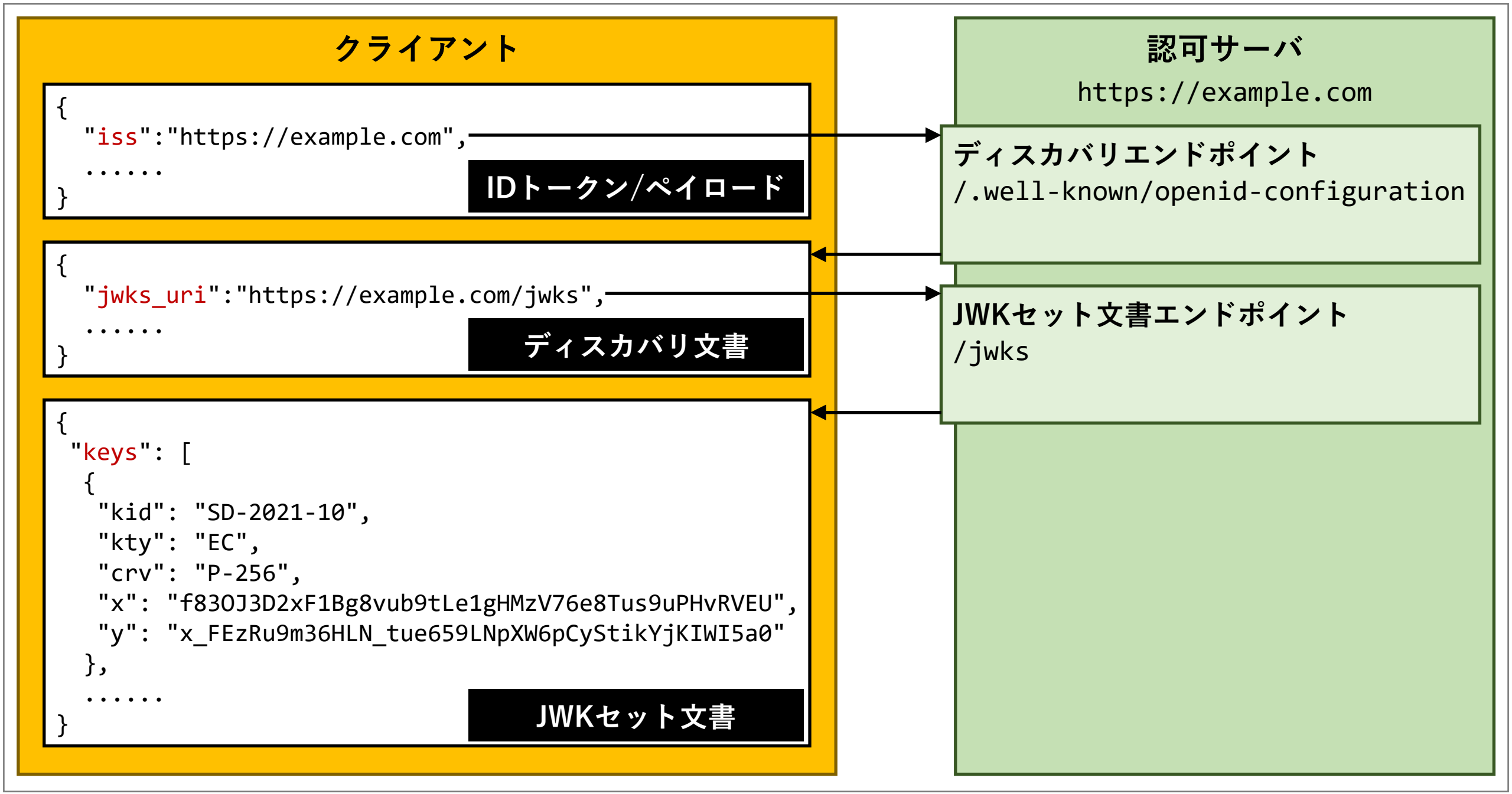
RFC 6750, Section 2.1 に挙げられている例

```
GET /resource HTTP/1.1
Host: server.example.com
Authorization: Bearer mF_9.B5f-4.1JqM
```

アクセストークン

curl コマンドで、アクセストークンを付けて Web API にアクセスする例

```
$ curl -H "Authorization: Bearer mF_9.B5f-4.1JqM"
  https://server.example.com/resource
```



IDトークン署名検証用鍵の取得

IDトークン署名検証のコード例

```
import com.nimbusds.jose.*;
import com.nimbusds.jose.crypto.*;
import com.nimbusds.jose.jwk.*;

public class SignatureVerificationExample
{
    private static final String ID_TOKEN   = "eyJ以下略";                    ←JWS形式のIDトークン
    private static final String PUBLIC_KEY = "{¥"kty¥":以下略";              ←JWK形式の公開鍵

    private static boolean verify(String jwsStr, String jwkStr) throws Exception
    {
        JWSObject   jws      = JWSObject.parse(jwsStr);                    ←文字列をJWSとしてパース
        JWK         jwk      = JWK.parse(jwkStr);                          ←文字列をJWKとしてパース
        ECKey       pub      = jwk.toECKey().toPublicJWK();                ←JWKを楕円曲線暗号公開鍵に変換
        JWSVerifier verifier = new ECDSAVerifier(pub);                     ←楕円曲線暗号署名の検証器を生成

        return jws.verify(verifier);                                       ←JWSの署名を検証
    }

    public static void main(String[] args) throws Exception
    {
        boolean result = verify(ID_TOKEN, PUBLIC_KEY);
        System.out.println("result = " + result);
    }
}
```

OpenID Connect Core 1.0, 3.1.3.7. ID Token Validation

Clients MUST validate the ID Token in the Token Response in the following manner:

1. If the ID Token is encrypted, decrypt it using the keys and algorithms that the Client specified during Registration that the OP was to use to encrypt the ID Token. If encryption was negotiated with the OP at Registration time and the ID Token is not encrypted, the RP SHOULD reject it.
2. The Issuer Identifier for the OpenID Provider (which is typically obtained during Discovery) MUST exactly match the value of the iss (issuer) Claim.
3. The Client MUST validate that the aud (audience) Claim contains its client_id value registered at the Issuer identified by the iss (issuer) Claim as an audience. The aud (audience) Claim MAY contain an array with more than one element. The ID Token MUST be rejected if the ID Token does not list the Client as a valid audience, or if it contains additional audiences not trusted by the Client.
4. If the ID Token contains multiple audiences, the Client SHOULD verify that an azp Claim is present.
5. If an azp (authorized party) Claim is present, the Client SHOULD verify that its client_id is the Claim Value.
6. If the ID Token is received via direct communication between the Client and the Token Endpoint (which it is in this flow), the TLS server validation MAY be used to validate the issuer in place of checking the token signature. The Client MUST validate the signature of all other ID Tokens according to **JWS** [JWS] using the algorithm specified in the JWT alg Header Parameter. The Client MUST use the keys provided by the Issuer.
7. The alg value SHOULD be the default of RS256 or the algorithm sent by the Client in the id_token_signed_response_alg parameter during Registration.
8. If the JWT alg Header Parameter uses a MAC based algorithm such as HS256, HS384, or HS512, the octets of the UTF-8 representation of the client_secret corresponding to the client_id contained in the aud (audience) Claim are used as the key to validate the signature. For MAC based algorithms, the behavior is unspecified if the aud is multi-valued or if an azp value is present that is different than the aud value.
9. The current time MUST be before the time represented by the exp Claim.
10. The iat Claim can be used to reject tokens that were issued too far away from the current time, limiting the amount of time that nonces need to be stored to prevent attacks. The acceptable range is Client specific.
11. If a nonce value was sent in the Authentication Request, a nonce Claim MUST be present and its value checked to verify that it is the same value as the one that was sent in the Authentication Request. The Client SHOULD check the nonce value for replay attacks. The precise method for detecting replay attacks is Client specific.
12. If the acr Claim was requested, the Client SHOULD check that the asserted Claim Value is appropriate. The meaning and processing of acr Claim Values is out of scope for this specification.
13. If the auth_time Claim was requested, either through a specific request for this Claim or by using the max_age parameter, the Client SHOULD check the auth_time Claim value and request re-authentication if it determines too much time has elapsed since the last End-User authentication.

アクセストークン実装の分類

RFC 6749, 1.4. Access Token より抜粋

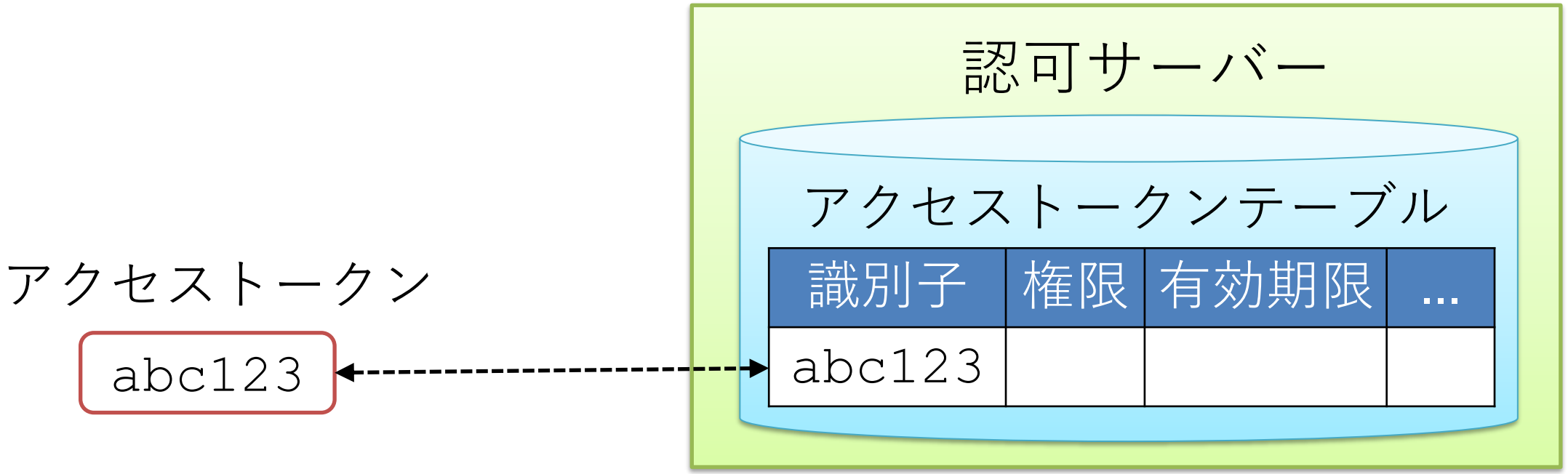
> *The token may denote an identifier used to retrieve the authorization information or may self-contain the authorization information in a verifiable manner (i.e., a token string consisting of some data and a signature). ...*

- ☑ アクセストークンの実装方法はおおむね『識別子型』と『内包型』の二つに大別することができる。
- ☑ これらを組み合わせる『ハイブリッド型』の実装もある。

識別子型

識別子型の実装では、アクセストークンに紐付く情報を認可サーバーのデータベース内に保存する。そして、そのデータレコードを一意に特定できる識別子をアクセストークンとする。

※ セキュリティのため、実際にはアクセストークンそのものではなくハッシュ値を保存することになる。



内包型

内包型の実装では、アクセストークンに紐付く情報をアクセストークン自体の中に埋め込む。

アクセストークン

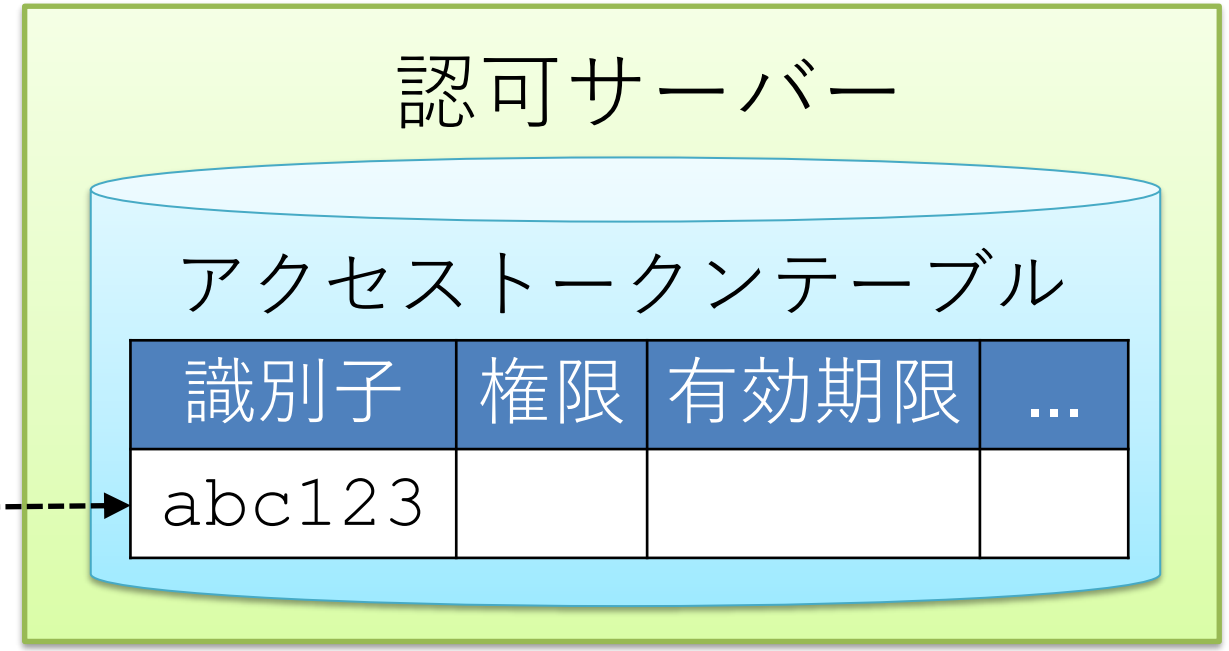
```
{
  "scope": "...",
  "client_id": "...",
  "exp": ...,
  "iat": ...,
  "sub": "...",
  "iss": "...",
  "jti": "..."
}
```

ハイブリッド型

ハイブリッド型の実装では、内包型アクセストークンを生成しつつも、それに付随するデータを認可サーバーのデータベース内に持つ。

アクセストークン

```
{
  "scope": "...",
  "client_id": "...",
  "exp": ...,
  "iat": ...,
  "sub": "...",
  "iss": "...",
  "jti": "..."
}
```

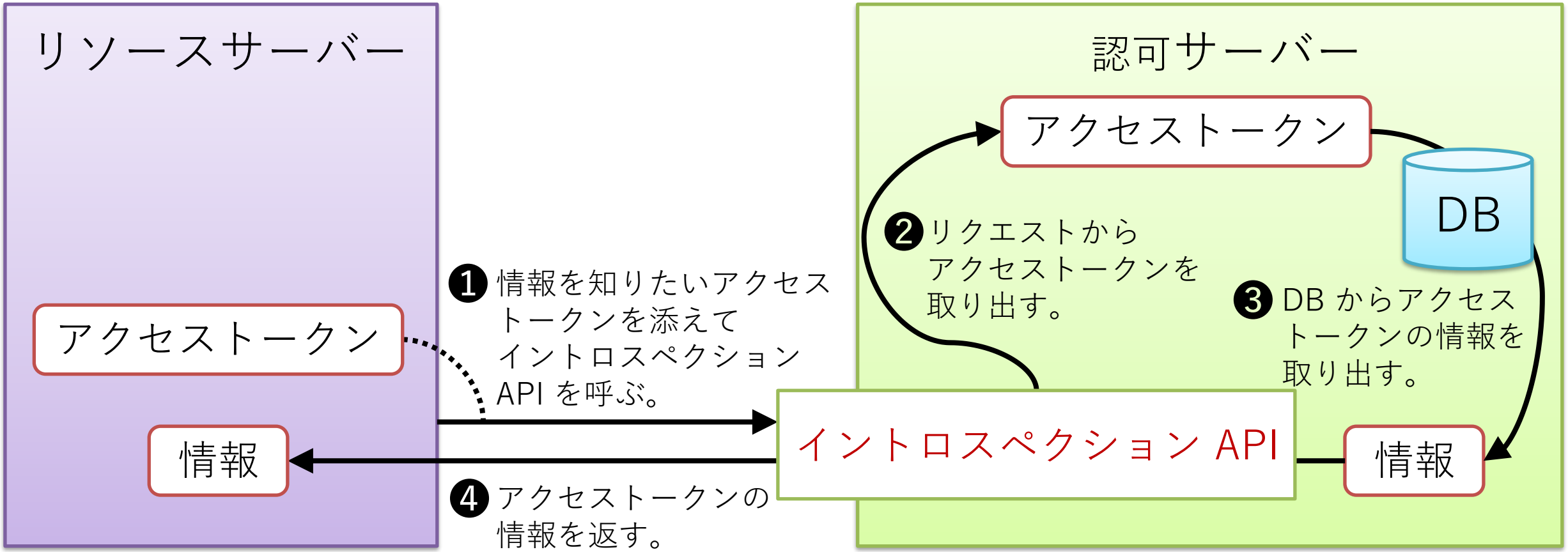


アクセストークン情報取得

識別子型アクセストークンの情報取得方法

認可サーバーが用意するイントロスペクション API に問い合わせる。

※ 認可サーバーとリソースサーバーが DB を共有しているシステムは、直接 DB を参照してアクセストークンの情報を得る。



RFC 7662 : OAuth 2.0 Token Introspection （イントロスペクション API の標準仕様）

イントロスペクションリクエスト（RFC 7662, Section 2.1）

HTTP メソッド	POST		
Content-Type	application/x-www-form-urlencoded		
パラメーター	token	必須	アクセストークンまたはリフレッシュトークン
	token_type_hint	任意	access_token または refresh_token

token_type_hint : *“The protected resource MAY pass this parameter to help the authorization server optimize the token lookup. If the server is unable to locate the token using the given hint, it MUST extend its search across all of its supported token types.”*

token_type_hint と実際に渡されたトークンの種類が異なっていても、エラーにならないばかりか、検索は続行される。

イントロスペクションレスポンス（RFC 7662, Section 2.2）

Status Code	200 OK		
Content-Type	application/json		
パラメーター	active	必須	true または false
	scope	任意	スコープ（RFC 6749, Section 3.3）
	client_id	任意	クライアントID
	username	任意	リソースオーナーの Human-Readable 識別子
	token_type	任意	トークンタイプ（RFC 6749, Section 5.1）
	exp	任意	有効期限終了時刻
	iat	任意	発行時刻
	nbf	任意	有効期限開始時刻
	sub	任意	主体識別子。通常はリソースオーナー識別子
	aud	任意	トークン利用者
	iss	任意	トークン発行者
	jti	任意	トークン識別子

アクセストークン失効


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

識別子型アクセストークンの失効　→ データベースからレコードを削除するだけ
内包型アクセストークンの失効　　→ PKI の CRL や OCSP 相当の仕組みを運用

CRL (Certificate Revocation List)

期限切れ前に失効した証明書群のシリアル番号一覧。
難点：リストが肥大化しうる。リスト同期の方法とタイミング。

OCSP (Online Certificate Status Protocol)

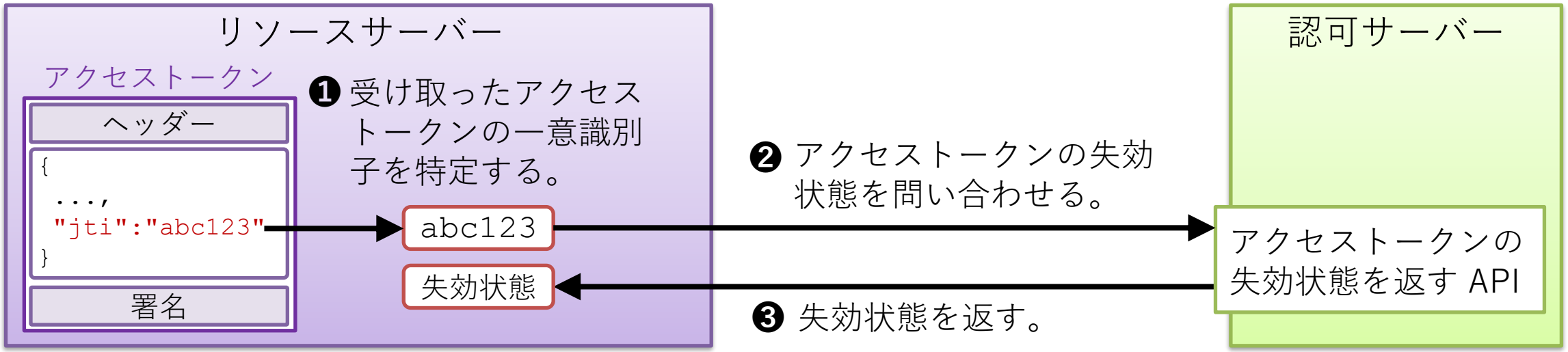
単一の証明書の失効状態をリアルタイムに問い合わせる仕組み。
難点：通信発生によるパフォーマンス問題。新たな単一障害点。

① CRL や OCSP 相当の仕組みを構築するためには、アクセストークンを一意に識別できなければならない。→ JWT の `jti` クレームを使えば実現可能。

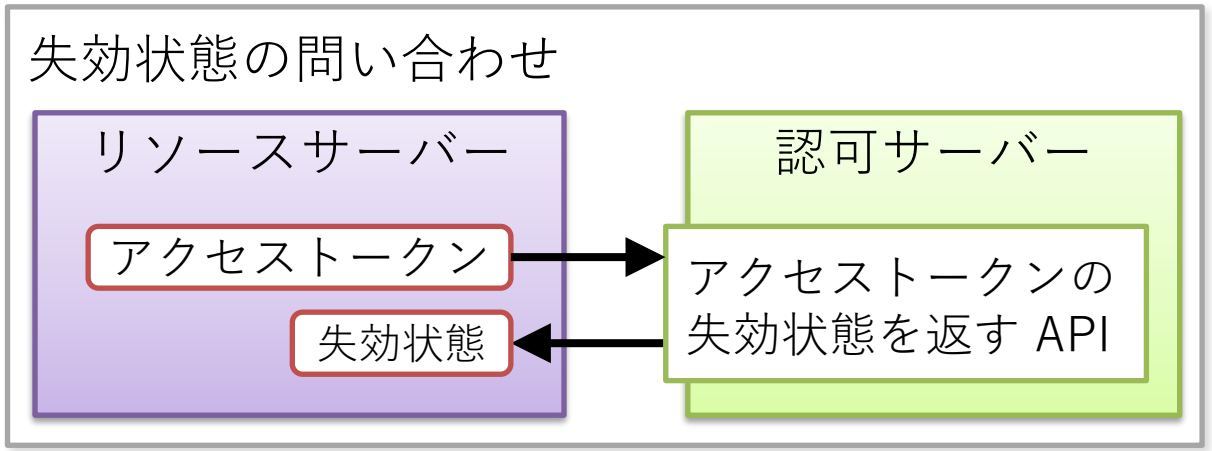
② アクセストークンを失効させるたびに、一意識別子を『失効されたアクセストークンのリスト』に追加する。当該アクセストークンの元々の有効期限が切れるまでは識別子をリスト内に保持しておく。

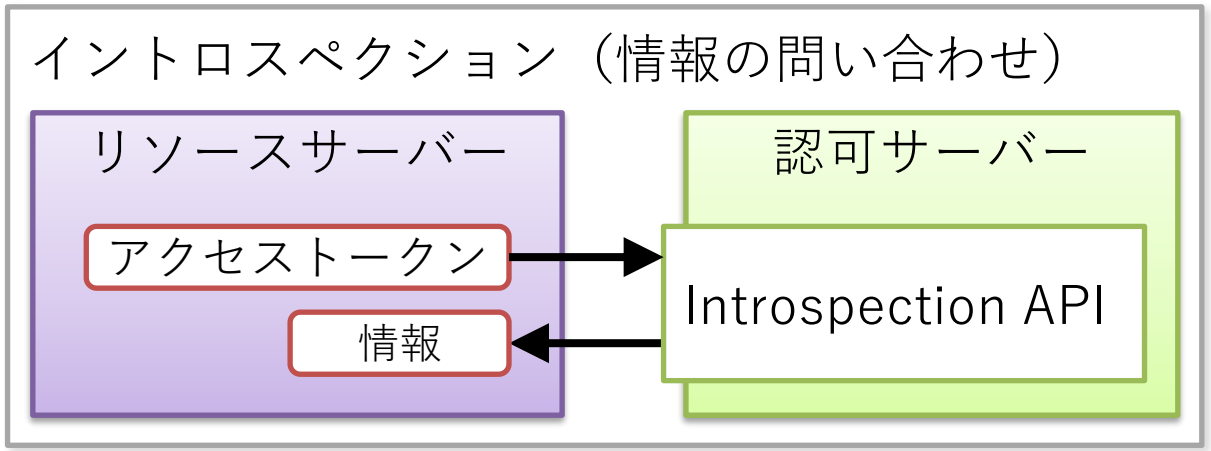
③ リソースサーバーは、受け取ったアクセストークンの失効状態を確認する。

CRL 相当	失効アクセストークンリストを取得し、当該一意識別子が含まれていないか確認
OCSP 相当	OCSP レスポンダに相当する『アクセストークン失効状態 API』に問い合わせ



失効状態の確認のために認可サーバーに問い合わせをすると、識別子型アクセストークンにおいてイントロスペクション API に問い合わせるのと同様に、ネットワーク通信が発生する。内包型アクセストークンの最大の利点が損なわれる。





というか、イントロスペクションレスポンス内の `active` に注目すれば、イントロスペクション API は『アクセストークンの失効情報を返す API』そのものであることが分かる。

よって、識別子型アクセストークンのほうが利点が多い（後述）ことを考慮すると、失効状態の問い合わせをしているようでは、内包型アクセストークンを積極的に採用する理由はほとんどない。

「アクセストークンの有効期間を短くして失効を諦める」という妥協をしない限り、内包型の利点はほとんどない。

これを差し置いても内包型を採用しなければならない理由があるとすれば、それは「何らかの事情によりリソースサーバーと認可サーバーが通信できない」場合のみ。

RFC 7009 : OAuth 2.0 Token Revocation

アクセストークン/リフレッシュトークンを
失効させるための API

リボケーションリクエスト（RFC 7009, Section 2.1. Revocation Request）

HTTP メソッド	POST		
Content-Type	application/x-www-form-urlencoded		
パラメーター	token	必須	アクセストークンまたはリフレッシュトークン
	token_type_hint	任意	access_token または refresh_token

RFC 7009, Section 2.1 に挙げられている例

```
POST /revoke HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW

token=45ghiukldjahdnhzdauz&token_type_hint=refresh_token
```

※ クライアント認証が必要

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

OAuth & OIDC 勉強会 【入門編】

2020 年 3 月 17 日に開催した弊社勉強会のプレゼンテーション録画です。

OAuth & OIDC 勉強会 【アクセストークン編】

2020 年 4 月 22 日に開催した弊社勉強会のプレゼンテーション録画です。

OAuth & OIDC 勉強会 【認可リクエスト編】

2020 年 5 月 27 日に開催した弊社勉強会のプレゼンテーション録画です。

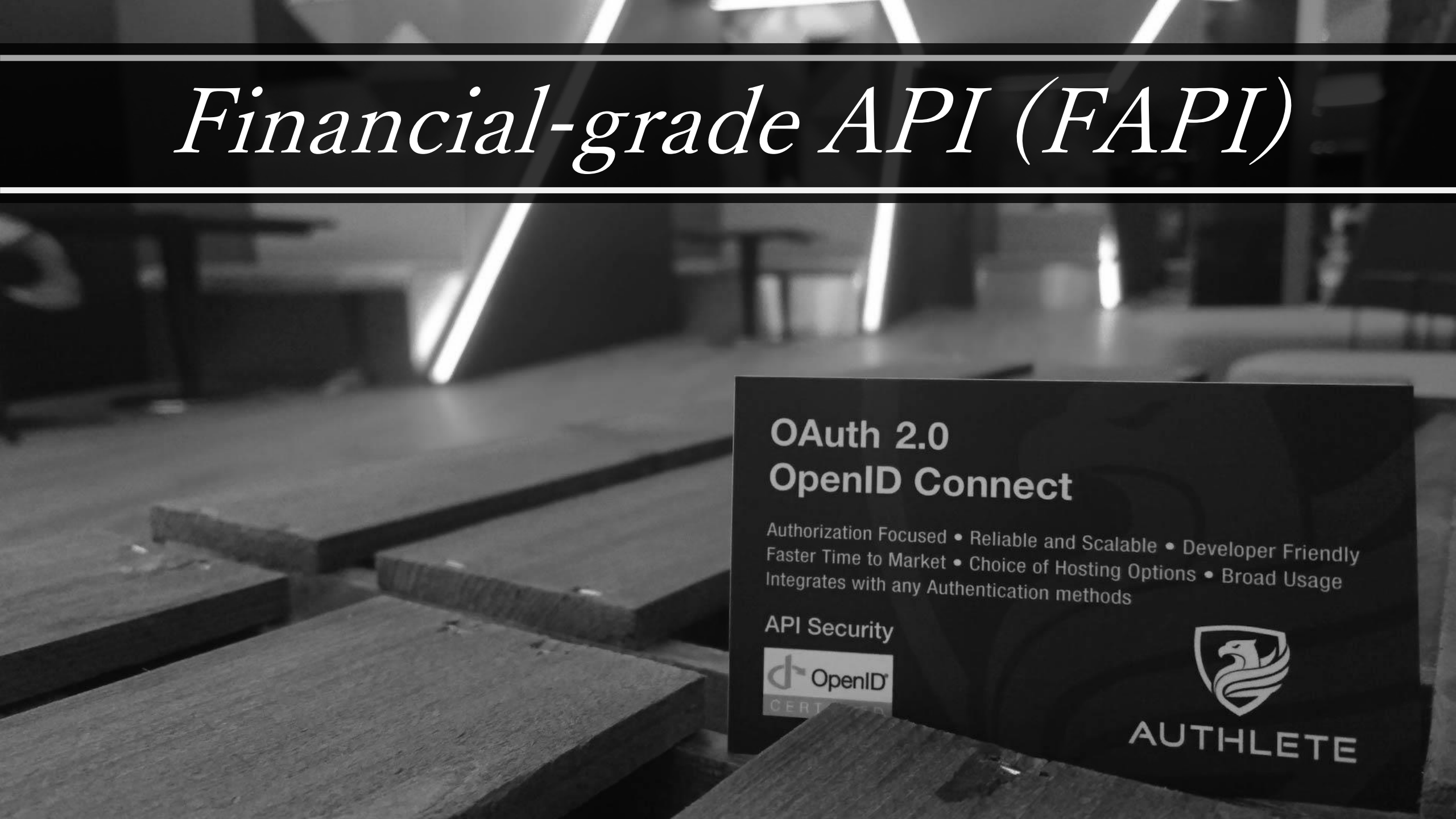
OAuth & OIDC 勉強会 【クライアント認証編】

2020 年 7 月 1 日に開催した弊社勉強会のプレゼンテーション録画です。

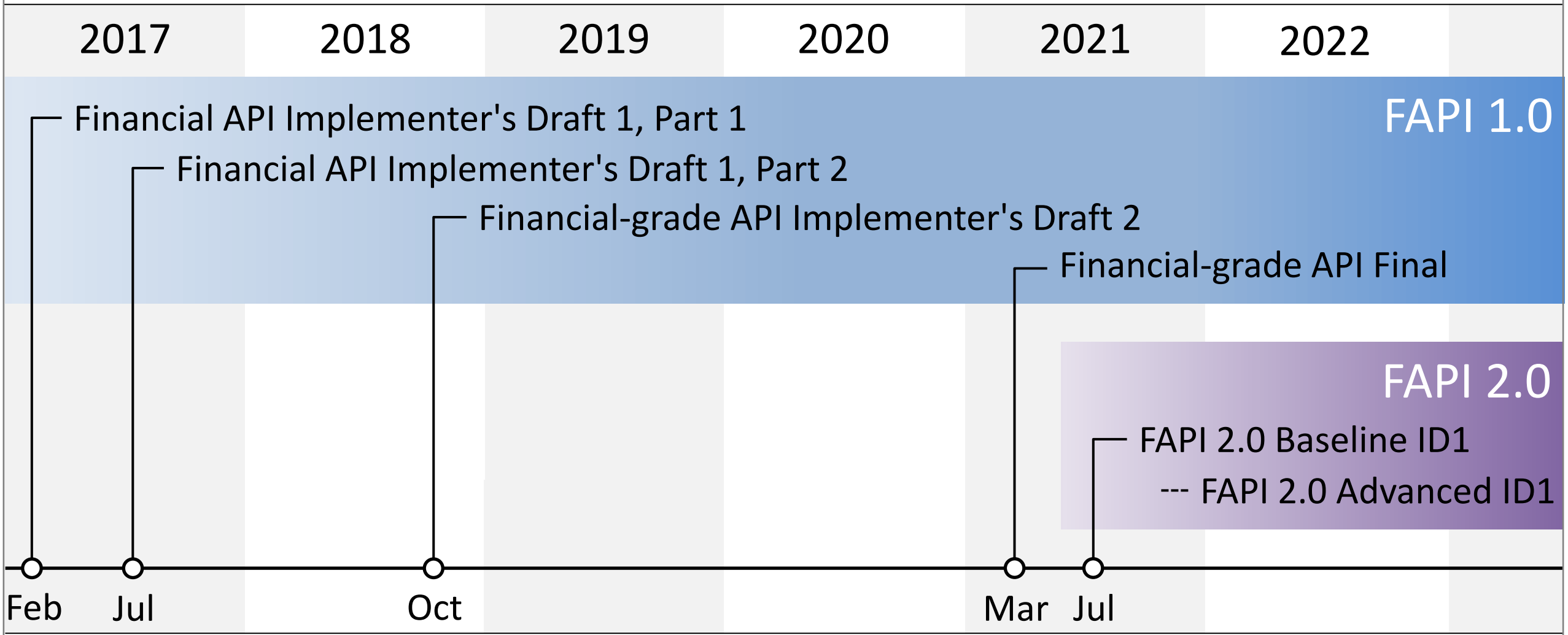
おまけ（最新動向）


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

Financial-grade API (FAPI)



Financial-grade API History



- "Financial API" was renamed to "Financial-grade API" by FAPI 1.0 Implementer's Draft 2.
- "Part 1: Read-Only Security Profile" was renamed to "Part 1: Baseline Security Profile" by FAPI 1.0 Final.
- "Part 2: Read and Write API Security Profile" was renamed to "Part 2: Advanced Security Profile" by FAPI 1.0 Final.

What is FAPI 2.0?

Specifications

FAPI 2.0 Baseline Profile, ID1	https://openid.net/specs/fapi-2_0-baseline-01.html
FAPI 2.0 Attacker Model, ID1	https://openid.net/specs/fapi-2_0-attacker-model-01.html

[announcement] "Implementer's Drafts of Two FAPI 2.0 Specifications Approved" on July 21, 2021
https://openid.net/2021/07/21/implementers-drafts-of-two-fapi-2-0-specifications-approved/

FAPI 2.0 Advanced Profile, ID1	Under discussion

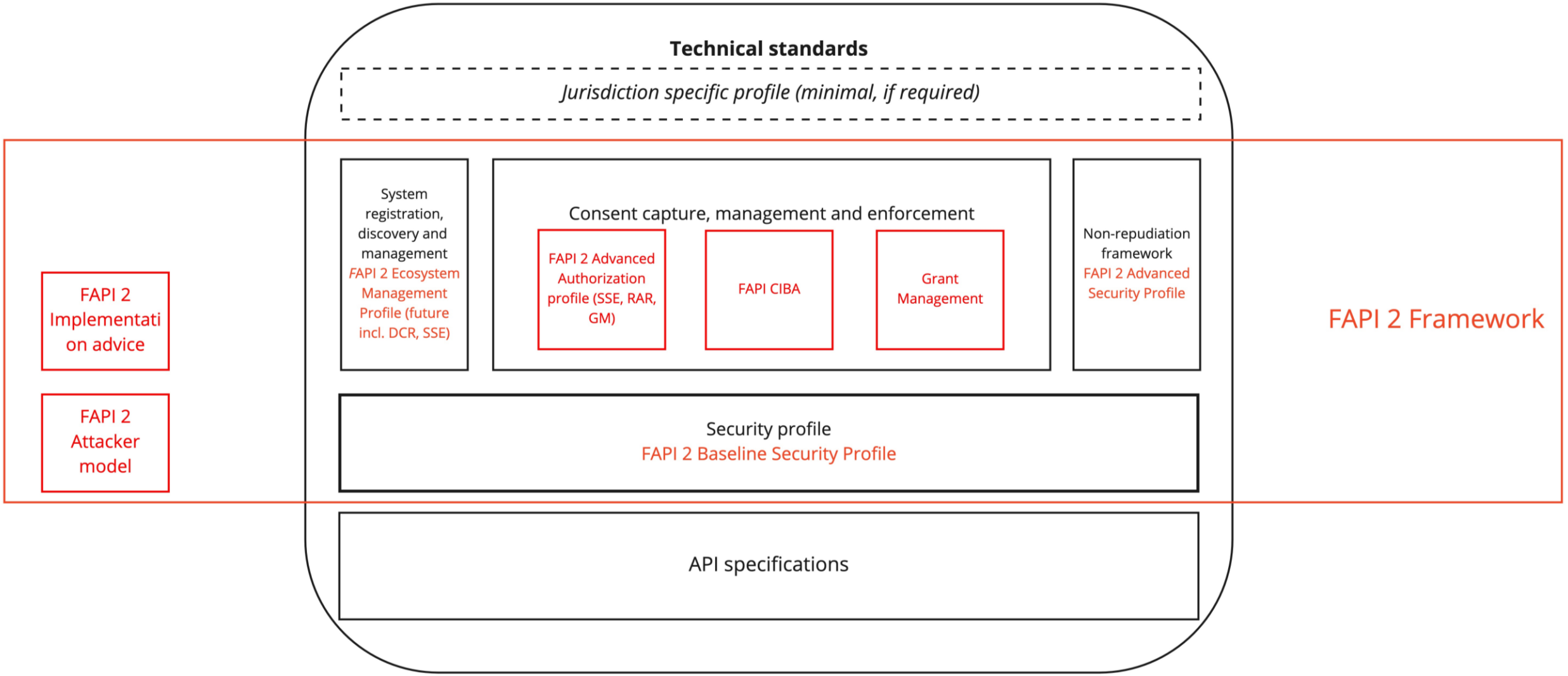
Tehcnical components listed in "FAPI FAQ" https://openid.net/wg/fapi/faq/

Component	Standard	Authlete	
PKCE	RFC 7636	1.1 ~	Proof Key for Code Exchange by OAuth Public Clients
mTLS	RFC 8705	2.0 ~	OAuth 2.0 Mutual TLS Client Authentication and Certificate Bound Access Tokens
DPoP	IETF Draft	2.2 ~	OAuth 2.0 Demonstration of Proof-of-Possession at the Application Layer (DPoP)
PAR	RFC 9126	2.2 ~	OAuth 2.0 Pushed Authorization Requests
RAR	IETF Draft	2.2 ~	OAuth 2.0 Rich Authorization Requests
iss	IETF Draft	2.2 ~	OAuth 2.0 Authorization Server Issuer Identifier in Authorization Response

FAPI Issue #432 FAPI2 Trust Framework structure

https://bitbucket.org/openid/fapi/issues/432/fapi2-trust-framework-structure

Discussion about how to position FAPI 2.0 itself and its technical components. The diagram posted by Dima on Aug 25, 2021 shows what "FAPI 2 Framework" includes and how the framework overlays ecosystem requirements.



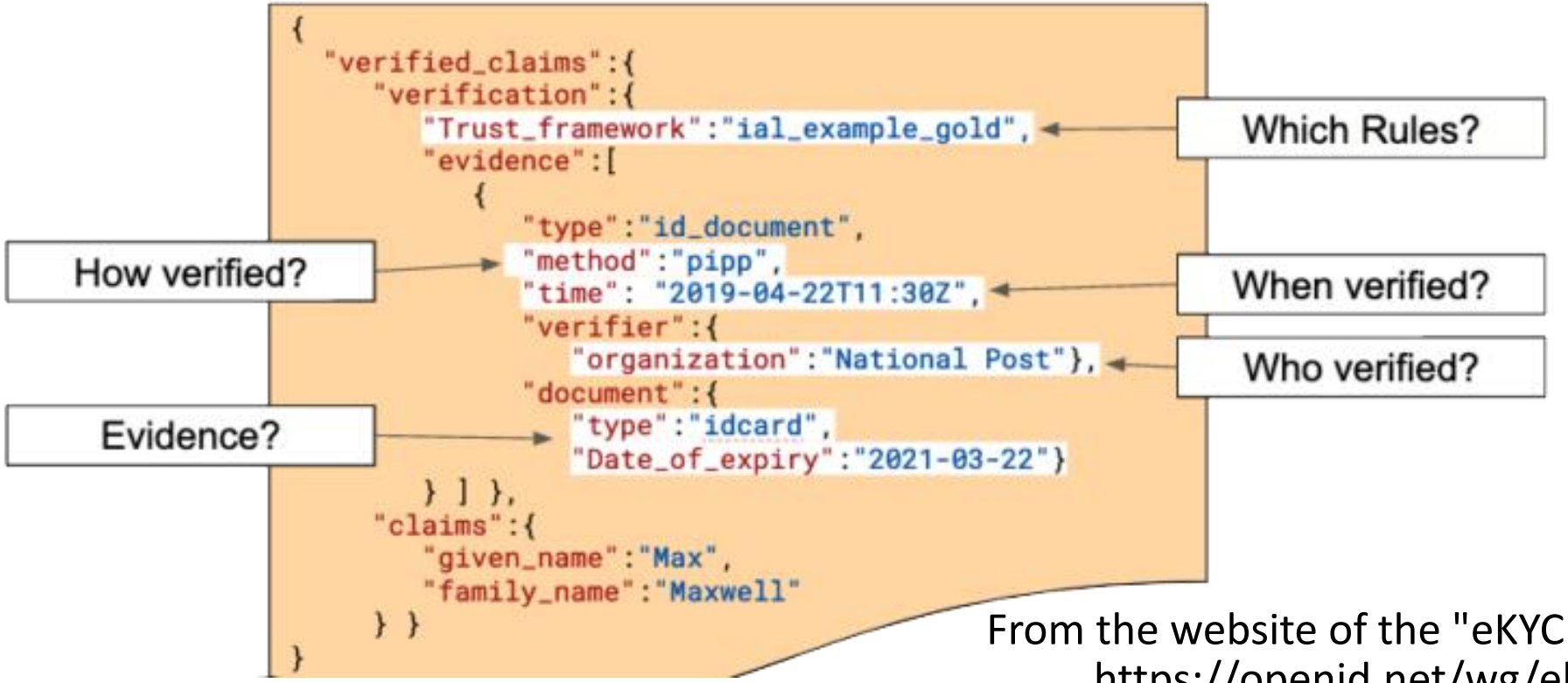
OpenID Connect for Identity Assurance (OIDC4IDA)

OpenID Connect for Identity Assurance 1.0 (OIDC4IDA)

A specification about how to request "verified claims" and how to represent them.

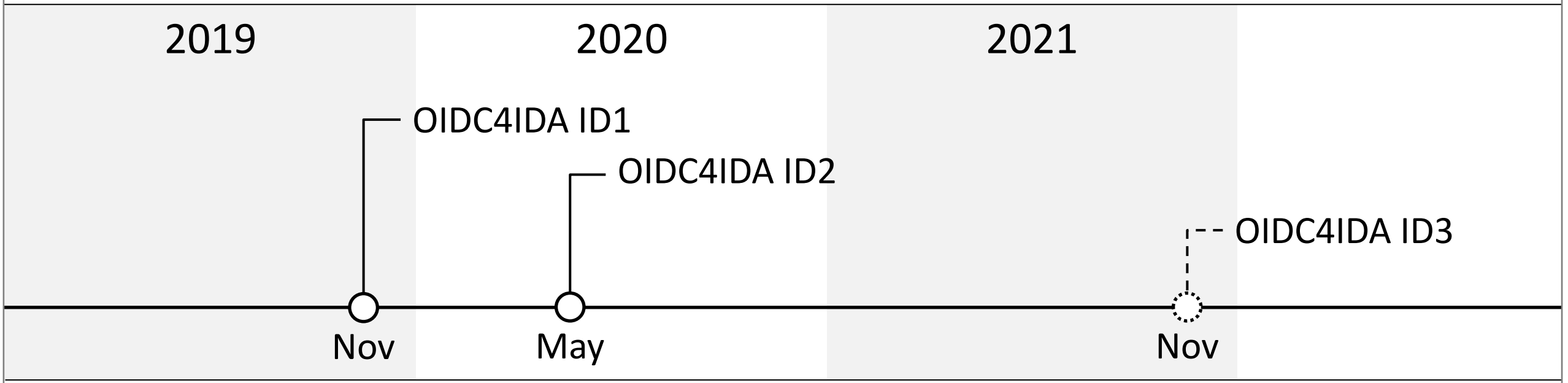
user attributes such as name and postal address

assured by official evidence such as passport and driver's license and by a trustworthy process such as face-to-face confirmation and eKYC.



From the website of the "eKYC and Identity Assurance" WG
https://openid.net/wg/ekyc-ida/

History of OpenID Connect for Identity Assurance 1.0



- The public review period of "OpenID Connect for Identity Assurance 1.0 Implementer's Draft 3" (OIDC4IDA ID3) started on September 18, 2021.
- The public review period will last 45 days. The specification will be approved at the beginning of November unless serious problems are found during the public review period.

Specification

OpenID Connect for Identity Assurance 1.0 Implementer's Draft 3 (under public review)

https://openid.net/specs/openid-connect-4-identity-assurance-1_0-12.html

→ To be implemented, but many additions and some changes since ID2.

Related Specifications to Watch

OpenID Connect Authority claims extension

https://bitbucket.org/openid/ekyc-ida/src/master/openid-connect-authority.md

OIDC4IDA is a specification to represent verified claims about a natural person while OIDC Authority claims a specification to represent a legal entitiy. A Japanese ministry, METI (Ministry of Economy, Trade, Industry; 経済産業省), shows interest in this specification.

OpenID Connect Advanced Syntax for Claims (ASC) 1.0

https://bitbucket.org/openid/ekyc-ida/src/master/openid-advanced-syntax-for-claims.md

A small query language for verified claims embedded in an authorization request. For example,

- `"if_different":"abort"`
- `"transformed_claims":{"above_18":{"claim":"birthdate","fn":["years_ago",["gte",18]]}}`

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