NRI SECURE

認証認可の調査研究

最終報告書

別添資料1 OpenID Foundation の策定規格 詳細

2020年09月25日

NRIセキュアテクノロジーズ株式会社

目次

1. OpenID Connect Core 1.0

OpenID Connect Core は、OpenID Connectの骨子となる仕様を定めている。現在の最新版は1.0であり、2014年11月に策定されている。

目次（青字でタイトルを記載している章は詳細を後述）

章番号	タイトル	概要
1	**Introduction**	OpenID Connect Core 1.0仕様の紹介。Open ID ConnectはOAuth2.0のプロトコル上にアイデンティティレイヤーを付与したことを述べている。
2	**ID Token**	IDトークンは、OpenID ConnectがOAuth2.0に追加する最大の拡張機能であり、ユーザの認証情報を持ったトークンであることを述べている。
3	**Authentication**	OpenID Connectで定義している認証フローである認可コードフロー、インプリシットフローおよびハイブリットフローについての説明を行っている。
4	Initiating Login from a Third Party	ログイン主体がRPからでなく、3rd Partyから開始する際のフローについて規定している。
5	**Claims**	クライアントがエンドユーザと認証に関するイベントで、ユーザ属性の種類とを取得方法について定義している。
6	Passing Request Parameters as JWTs	認可要求に署名及び暗号化を可能にするための仕様を定義している。
7	Self-Issued OpenID Provider	自己署名つきの ID トークンを発行しself-hostedなOPをサポートするための仕様について規定している。
8	Subject Identifier Types	Subject Identifierはクライアントによって利用されるエンドユーザの識別子であり、その仕様について定義されている。
9	**Client Authentication**	クライアントがトークンエンドポイントにアクセスする際に、認可サーバで自身を認証するための方法を定義している。

OpenID Connect Core は、OpenID Connectの骨子となる仕様を定めている。現在の最新版は1.0であり、2014年11月に策定されている。

目次（続き）

章番号	タイトル	概要
10	**Signatures and Encryption**	JSON Web Signature(JWS)とJSON Web Encryption(JWE)をつかって、コンテンツの暗号化と署名を行うための仕様について記載されている。
11	Offline Access	OAuth2.0 リフレッシュトークンの発行要求するためのscope値に関して定義している。
12	**Using Refresh Tokens**	認可サーバが、リフレッシュトークンを受け取った際の挙動について定義されている。
13	Serializations	メッセージをシリアライズする方法の構文について定義されている。
14	String Operations	OpenID Connectのメッセージを処理する際に、メッセージの中の既知の値を比較することが求められるが、その際の手順について定義されている。
15	**Implementation Considerations**	RPとOPの両者によって使用される機能について定義されている。
16	**Security Considerations**	RFC6750、6819、ISO/IEC29115を参照して、実装者が考慮すべきセキュリティ事項に関して説明されている。
17	**Privacy Considerations**	OpenID Connectのフローにおいて、個人情報保護の観点から考慮すべき事項について述べられている。
18	IANA Considerations	OpenID Connectで定義しているJWTのクレーム、OAuthのパラメータ、エラー種別を、JWT及びRFC6749に登録するということが述べられている。

OpenID Connect Core 1.0中に関する主な用語

用語	規定されている ドキュメント	概要
OpenID Provider (OP)	OpenID Connect Core 1.0	ユーザの認証およびIDトークン、アクセストークンの発行機能を有するサーバ。ユーザ認証後にRPからの要求に対してIDトークンとアクセストークンの発行を行う。
Relying Party (RP)	OpenID Connect Core 1.0	OPに対してユーザの認証とユーザ情報を要求するOAuth2.0 クライアント。 OPから認証結果を受け取り、OPに対してアクセストークンとIDトークンの発行を要求し、アクセストークンを使ってUserInfoエンドポイントに対して要求を行い、アイデンティティ情報を入手する。
IDトークン	OpenID Connect Core 1.0	ユーザ認証に関する情報を含むトークン。OPによって発行され、エンドユーザの認証に使われる。ID Tokenの形式は署名付きのJSON Web Token（JWT)。
リソースサーバ	OAuth 2.0	アクセストークンを使用して保護されたリソースに対する要求を受け入れて応答することができるサーバ。
アクセストークン	OAuth 2.0	RPがリソースサーバにアクセスするために使われるトークン。
UserInfoエンドポイント	OpenID Connect Core 1.0	RPからの要求に対して、エンドユーザのプロフィール情報が提供されるエンドポイント。
JWT	OpenID Connect Core 1.0	JSON Web Tokenの略称で、属性情報（Claim）をJSONの構造で表現したトークン。
Issuer	OpenID Connect Core 1.0	IDトークンの発行者を示すURL形式の値。
Discovery	OpenID Connect Discovery 1.0	OPは、OpenID Connect Discovery 1.0の仕様をサポートすることで、OPの発行者識別子（https://から始まるURI）に” /.well-known/openid-configuration”を付加したURLにGETリクエストすることでOPの情報を取得することができる。
クライアント	OAuth 2.0	保護されたリソースの所持者から認可を得て、リソースオーナーの代理として保護されたリソースに対するリクエストを行うアプリケーション。
クライアントID	OAuth 2.0	クライアントの識別子。
クライアントシークレット	OAuth 2.0	クライアントだけが知っている秘密の値。ユーザ認証の際のID/PWのパスワードに相当。

ID連携機能を提供するOpenID 2.0と、API連携のためのアクセス制御機能を提供するOAuth2.0を統合することでOpenID Connectの仕様が策定された。

ID連携

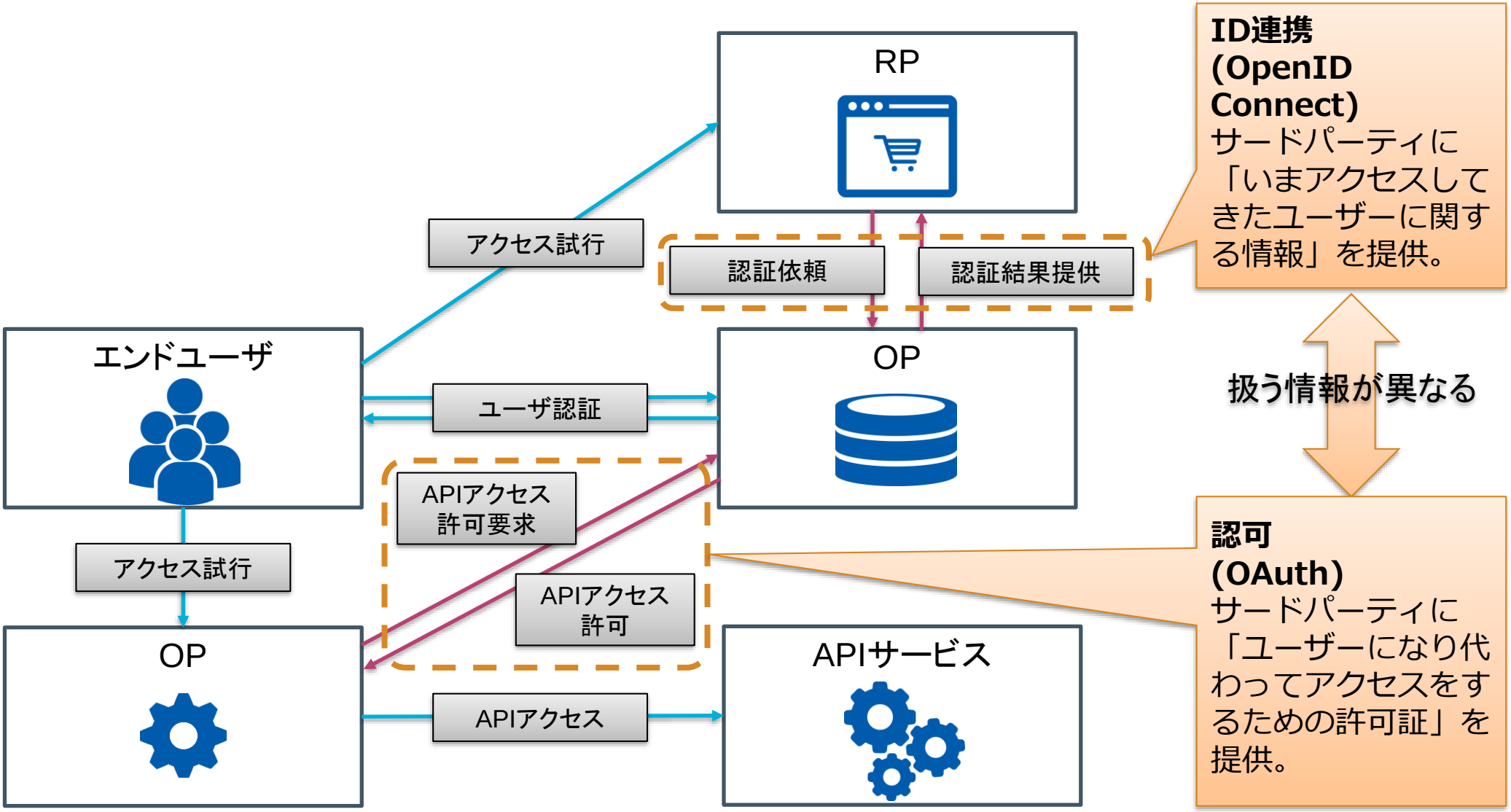
ID情報（認証結果と属性情報）を、安全にサービス間でやりとりするための仕様

API連携

あるサービスのAPIアクセスの許可を、別のサービスに安全に与えるための方法

OpenID 2.0 + OAuth 2.0 = OpenID Connect

OpenID ConnectとOAuthはシーケンスなど類似する点があるものの、提供する機能および扱う情報が異なる。



IDトークンは、OpenID ConnectがOAuth2.0に追加する最大の拡張機能。OPからRPに対して発行されエンドユーザの認証に関する情報を含む。

IDトークン

- OPからRPに対して発行されて、エンドユーザの認証に関する情報を含む。
- JWT形式で表現されており、JWS(JSON Web Signature)を使って署名される。
- クレームと呼ばれる認証に関する各種パラメタとして以下の情報を含む。

クレーム	条件	説明
iss	Required	IDトークンの発行者を示すURL形式。
sub	Required	Issuerごとにユニークで再利用されないエンドユーザの識別子。
aud	Required	Audienceの略で、IDトークンの発行を受けるRPのクライアントID。
exp	Required	IDトークンの有効期限。
iat	Required	JWTの発行時刻。
auth_time	条件付き	エンドユーザの認証をした時刻。リクエストに max_age が含まれていた場合必須。
nonce	条件付き	クライアントIDセッションとIDトークンを紐づける文字列値。リプレイアタック防止のために使われる。認可リクエストに含まれていた場合は必須。
acr	Optional	ユーザー認証が満たした認証コンテキストのクラスの値を示す文字列。
amr	Optional	認証時に用いられた認証方式を示す識別子文字列のJSON配列。
azp	Optional	IDトークン発行対象である認可されたクライアントID。発行依頼したクライアントと認可されたクライアントが違う場合必要。

OpenID ConnectではRPの種別に応じた3種類のフローをサポートしているが、一般的には認可コードフローが利用されている。

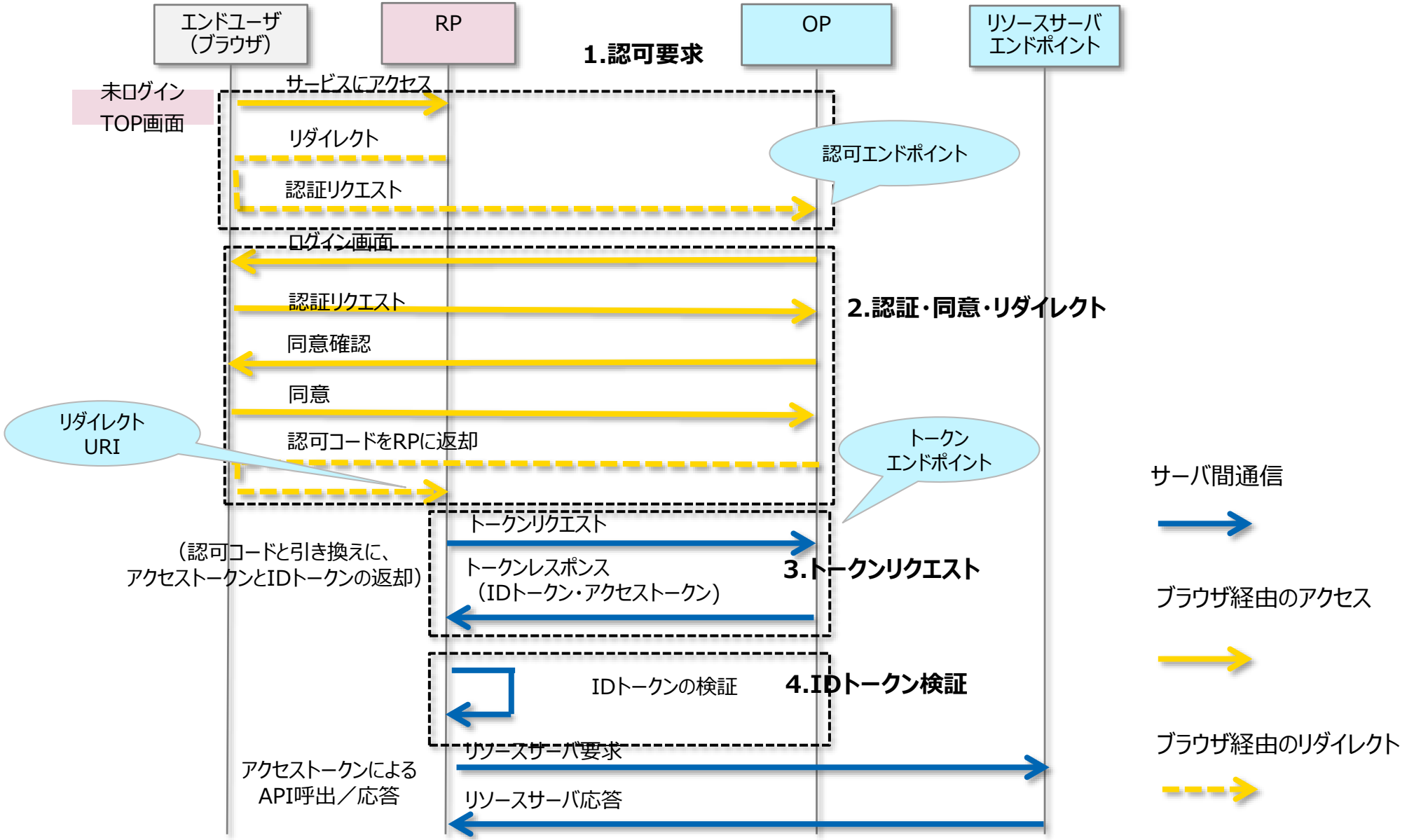
OpenID Connectでサポートするフロー

- **認可コードフロー**
  - 認可コードフローではクライアントに認可コードを返却し、 クライアントはそれを直接 IDトークンおよびアクセストークンと交換を行う。これにより、 User AgentやUser Agent上の他の不正アプリケーションなどにトークンも露呈することを防ぐことができる。
  - サーバサイドアプリからリソースサーバにアクセスする際に用いられる。
- **インプリシットフロー**
  - JavaScriptなどスクリプト言語を使用してブラウザに実装されたクライアントに対して、アクセストークンとIDトークンが直接返却されるフロー。すべてのトークンは認可エンドポイントから返され、トークンエンドポイントは使用されない。
  - フロントサイドアプリからリソースサーバにアクセスする際に用いられる。
  - アクセストークンとID トークンが、エンドユーザとエンドユーザのUser Agentにアクセスするアプリに露出する可能性がある。
- **ハイブリットフロー**
  - ハイブリットフローでは、 いくつかのトークンは認可エンドポイントから返され、その他のトークンはトークンエンドポイントから返される。
  - モバイル端末側のネイティブアプリとWebサーバー側のバックエンドアプリケーションからなる構成の際に、用いられる。

フロー選択に関する注意事項

- インプリシットフローは、アクセストークン・IDトークンを露出する可能性があり、特別な事情が無い限りは認可コードフローの利用が推奨される。
- 上記の点より、以降では認可コードフローの仕様についてのみ詳細を記載する。

OpenID Connect 認可コードフローの処理シーケンス



認可コードフロー： 1. 認可要求

認可要求時のパラメータ

- 認可コードフローシーケンス図の「1. 認可要求」のシーケンスに相当。
- エンドユーザがサービスにアクセスした際に、下記のようなリダイレクトレスポンスが返され、認可コードフローのシーケンスが開始される。

```
HTTP/1.1 302 Found
Location: https://server.example.com/authorize?
response_type=code
&scope=openid%20profile%20email
&client_id=s6BhdRkqt3
&state=af0ifjsldkj
&redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb
&nonce=n-0S6_WzA2Mj
```

項目	条件	説明
response_type	Required	シーケンスの種類。”code”は認可コードフローであることを示す。
scope	Required	RPが要求するOPに要求するscope。
client_id	Required	OPにRPを登録した際に発行されるクライアントを識別するID。
state	Recommended	リクエストとコールバックの間で維持されるランダムな値。 エンドユーザとのセッションに紐づけてCSRFやXSRFを防ぐ。
redirect_uri	Required	レスポンスが返されるリダイレクトURI。OPに対して事前登録済みのリダイレクトURIと完全一致しなければならない。
nonce	Optional	クライアントセッションと、IDトークンを紐づける文字列値。

認可コードフロー： 2. 認証・同意・リダイレクト

認証・同意・リダイレクト

- 認可コードフローシーケンス図の「2.認証・同意・リダイレクト」のシーケンスに相当。
- 認証
  - 認可要求した後に、OPのログイン画面がエンドユーザに表示されユーザ認証が求められる。
- 同意
  - 認証に成功すると認可サーバはRPにエンドユーザの情報を送る前にエンドユーザの同意を取得する必要がある。
  - 一般的に、認可要求時のscope値に応じてエンドユーザに対してRPにユーザ情報提供もしくはユーザ権限の委譲をしてよいか同意が求められる。（同意と取得方法に関しては定義されていない）
- リダイレクト
  - 同意が完了するとOPがリダイレクトレスポンスを返して、locationヘッダーには認可要求時のredirect_uriがセットされる。

```
HTTP/1.1 302
Found Location: https://client.example.org/cb?
code=SplxlOBeZQQYbYS6WxSbIA
&state=af0ifjsldkj
```

パラメータ	説明
code	認可コード。後にRPがOPに対してアクセストークンとIDトークンを要求する際に使われる。
state	クエストとコールバックの間で維持されるランダムな値。エンドユーザとのセッションに紐づけてCSRF(XSRF)を防ぐ。

認可コードフロー： 3. トークンリクエスト

トークンリクエスト

- 認可コードフローシーケンス図の「3. トークンリクエスト」のシーケンスに相当。
- RPからトークンエンドポイントに対してリクエストが行われる。
- Basic認証によりクライアントID、クライアントシークレットを送付する必要がある。また、POSTのBody部に、クライアントID、クライアントシークレットを含める方式も認められている。（クライアントがpublic clientの場合は、クライアントIDのみでよい）

```
POST /token HTTP/1.1 Host:

server.example.com Content-Type:
application/x-www-form-urlencoded
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
grant_type=authorization_code
&code=SplxlOBeZQQYbYS6WxSbIA
&redirect_uri=https%3A%2F%2Fclient.example.org%2Fc
b
```

パラメータ	条件	説明
grant_type	Required	認可コードフローであるため、”authorization_code”を設定する。
code	Required	リダイレクトレスポンスに含まれていたcode値を設定する。
redirect_uri	Required	認可要求時にリクエストに含めたredirect_uriと同じ値を設定する。

認可コードフロー： トークンレスポンス

トークンレスポンス

- トークンリクエストの検証に成功すると、トークンエンドポイントよりIDトークンとアクセストークンを含んだレスポンスが返却される。
- IDトークンは[.]区切りで、”{ヘッダー部}.{ペイロード部}.{署名部}”から構成されており、JSON Web Signature（JWS）と呼ばれる形式となる。

パラメータ	説明
access_token	アクセストークン。
token_type	トークンの種別。ここではトークン種別が、”Bearer”であることを示す。
refresh_token	リフレッシュトークン。
expires_in	アクセストークンの有効期限(秒)。
id_token	IDトークン。

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache
{
 "access_token": "SlAV32hkKG"、
 "token_type": "Bearer"、
 "refresh_token": "8xLOxBtZp8"、
 "expires_in": 3600、
 "id_token":
"eyJhbGciOiJSUzI1NiIsImtpZCI6IjFlOWdkazcifQ.ewogImlzc
 yI6ICJodHRwOi8vc2VydmVyLmV4YW1wbGUuY29tIiwKICJzdWIiOiAiMjQ4Mjg5
 NzYxMDAxIiwKICJhdWQiOiAiczZCaGRSa3F0MyIsCiAibm9uY2UiOiAibi0wUzZ
 fV3pBMk1qIiwKICJleHAiOiAxMzExMjgxOTcwLAogImlhdCI6IDEzMTEyODA5Nz
 AKfQ.ggW8hZ1EuVLuxNuuIJKX_V8a_OMXzR0EHR9R6jgdqrOOF4daGU96Sr_P6q
 Jp6IcmD3HP99Obi1PRs-cwh3LO-p146waJ8IhehcwL7F09JdijmBqkvPeB2T9CJ
 NqeGpe-gccMg4vfKjkM8FcGvnzZUN4_KSP0aAp1tOJ1zZwgjxqGByKHiOtX7Tpd
 QyHE5lcMiKPXfEIQILVq0pc_E2DzL7emopWoaoZTF_m0_N0YzFC6g6EJbOEoRoS
 K5hoDalrcvRYLSrQAZZKflyuVCyixEoV9GfNQC3_osjzw2PAithfubEEBLuVVk4
 XUVrWOLrLl0nx7RkKU8NXNHq-rvKMzqg“
}
```

IDトークンはJSON形式のデータに電子署名が付加されたJWS形式となっており、セキュリティ要件に応じて暗号化にも対応している。

IDトークン詳細

- IDトークンはヘッダー部、ペイロード部、署名部で構成させるJWS（JSON Web Signature）形式。
- IDトークンのペイロード部はJWT形式（クレームをJSON構造で表現したトークン）となる。
- IDトークンはJWE（JSON Web Encryption）形式での暗号化にも対応している。
- IDトークンのヘッダー部とペイロード部は、それぞれbase64urlでエンコードされているため、base64urlでデコードすることで元のデータを得ることが可能。
- 署名部は、ヘッダーとペイロードに対して、base64urlでエンコードした後、ヘッダーに含まれている alg パラメーターの値を使って署名を行い、さらにbase64urlでエンコードを施す。

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

IDトークンのヘッダーに対して、
base64urlでデコードした結果

```
{
 "iss": "http://server.example.com",
 "sub": "248289761001",
 "aud": "s6BhdRkqt3",
 "nonce": "n-0S6_WzA2Mj",
 "exp": 1311281970,
 "iat": 1311280970
}
```

IDトークンのペイロードに対して、
base64urlでデコードした結果

認可コードフロー： 4. IDトークンの検証

IDトークンの検証

- 認可コードフローシーケンス図の「4.IDトークンの検証」のシーケンスに相当。
- IDトークンを受け取ると、RPは攻撃されることを防ぐため、ペイロードと署名の検証を行わなければならない。

	検証項目（一部抜粋）	説明
ヘッダー部	alg	デフォルトのRS256か、クライアントにより、id_token_signed_response_alg パラメータとして送られたアルゴリズムであるべき。
ペイロード部	iss	Discoveryを通して取得される、OPのIssuerと正確に一致しなければならない。
	aud	IDトークンが発行されるクライアントIDの値が記載されている。 自信のクライアントIDと一致しなければならない。
	iat	IDトークンが発行された時間が記載されている。 現在時刻からはるか昔に発行されていた場合、拒絶することができる。
	nonce	認可要求時のnonceと値が一致するか、検証しなければならない。
	azp	複数のaudienceを含むならば、azpクレームがあるか確認すべき。
	exp	現在時刻よりも前でなければならない。
	auth_time	auth_timeクレームが要求されたら、特定のリクエストまたはmax_ageパラメータを用いて、auth_timeクレームの値をチェックし、最新のユーザ認証から長い時間が経過してた場合には、再認証を要求すべき。
署名部	署名	IDトークンが改ざんされる恐れがあるため、署名の検証を行わなければならない。

OpenID Connectでは、エンドユーザの属性情報に関する各種クレームが標準的なセットとして定められている。

クレームの定義

- OpenID Connectのメッセージの中でやりとりされるエンドユーザの属性情報。
- 仕様上定められている各クレームとエンドユーザ属性情報の対応は下記の通り。

クレーム	説明
sub	エンドユーザの識別子。
name	エンドユーザのフルネーム。
given_name	エンドユーザの名。
family_name	エンドユーザの姓。
middle_name	エンドユーザのミドルネーム。
nick_name	エンドユーザのニックネーム。
preferred_username	エンドユーザの簡略名。
profile	エンドユーザのプロフィールページのURL。
picture	エンドユーザのプロフィール画像のURL。
website	エンドユーザのWebページやブログのURL。

クレーム	説明
email	エンドユーザのemailアドレス。
email_verified	エンドユーザのemailが検証済みならtrue、さもなくばfalse。
gender	エンドユーザの性別。
birthdate	エンドユーザの誕生日。
zoneinfo	エンドユーザのタイムゾーン。
locale	エンドユーザの言語コードを小文字表記、国コードを大文字でダッシュ(“-”)でつなげたもの。
phone_number	エンドユーザの電話番号。
phone_number_verified	エンドユーザの電話番号が検証済みならtrue、さもなくばfalse。
address	エンドユーザの住所。
updated_at	エンドユーザの情報に関する最終更新日時。

エンドユーザのクレームはUserInfoエンドポイントから提供され、認可要求時に指定したスコープと紐づけられたクレームが返却される。

UserInfoエンドポイント

- 認証されたエンドユーザに関するクレームを返すエンドポイント。
- クレームを得るためにはOpenID Connectの認証を通じて得られたアクセストークンを使いUserInfoエンドポイントにリクエストを行う必要がある。

スコープ

- 認可要求を行う際にどのクレームを取得したいのか指定するために、スコープの指定を行う。
- OpenID Connectでは以下のスコープが定義されている。
- 独自のスコープを定義し運用することも可能。

スコープ	クレーム
profile	name、 family_name、 given_name、gender、locale、zoneinfo等のプロフィール情報を返却
email	emailおよびemail verifiedを返却
address	addressを返却
openid	(OpenID Connectのリクエストであることを示し、IDトークンが返却される)
phone	phone_numberおよびphone_number_verifiedを返却

OpenID Connectでは、クレームの提供元に応じた3種類のクレームが定義されている。

クレーム種別

クレーム種別	説明
Normalクレーム	OPによって直接提供されたクレーム
Aggregatedクレーム	OP以外のクレームプロバイダーによって提供されているが、OPが集約し返却するクレーム
Distributedクレーム	OP以外のクレームプロバイダーに提供されており、OPからはその情報への参照だけが返却されるクレーム

Normalクレーム例

- JSONオブジェクトの中に、各クレームの値が表記される。

```
{
 "name": "Jane Doe"、
 "given_name": "Jane"、
 "family_name": "Doe"、
 "email": "janedoe@example.com"、
 "picture": "http://example.com/janedoe/me.jpg"
}
```

OpenID Connectでは、2種類のSubject Identifierが定義されている。

Subject Identifier 概要

- クライアントによって利用される、Issuer内で一意であり再割り当てされないエンドユーザの識別子。
- Subject identifierには、publicとpairwiseの2種類が規定されている。

種別	説明
Pairwise	各クライアントに対して**異なる識別子を発行**する。 クライアントにまたがった名寄せが困難になる。
Public	各クライアントに対して**同じ識別子を発行**する。 クライアントにまたがった名寄せが可能。

OpenID Connectでは、5種類のクライアント認証方式が定義されている。

クライアント認証の定義

- クライアントがトークンエンドポイントにアクセスする際に、認証サーバに対して自身を認証する方法。
- OpenID Connect Coreの仕様では下記の5つの方法が定義されている。

方式	説明
client_secret_basic	認可サーバから受け取ったクライアントシークレットを使い、HTTP Basic認証の利用をして、認可サーバに対して自身を認証する。
client_secret_post	認可サーバから受け取ったクライアントシークレットを使い、クライアントクレデンシャルをリクエストボディに含めて、認可サーバに対して自身を認証する。
client_secret_jwt	認可サーバから受け取ったクライアントシークレットを使いJWTを生成し、OAuth.JWTおよびOAuth.Assertionsに従い認証を行う。
private_key_jwt	公開鍵を登録済みのクライアントは、そのペアとなる秘密鍵でJWTの署名を行ってもよい。
none	クライアントは、トークンエンドポイントで認証を行わない。これは、インプリシットフローか、publicクライアントである場合が該当するが、その他のなんらかの認証手段を用いる場合がある。

OpenID Connectでは、各種トークンやレスポンスに対する署名および暗号化に対応している。

署名と暗号化

- メッセージの送信経路によってはメッセージの完全性が保証されないことがあり、またメッセージ送信者の認証も行われないことがある。
- 上記のリスクを軽減するため、以下のメッセージについてはJSON Web Signature (JWS)によって署名をすることが可能である。また、メッセージの隠匿性を得るためにJSON Web Encryption (JWE)による暗号化も可能である。
  - IDトークン
  - UserInfoレスポンス
  - 6章に定義されているリクエストオブジェクトおよびクライアント認証 JWT

定義されている署名・暗号化方式

- **署名方式**
  - 非対称署名（RSA、ECDSA）
  - 対称署名（MACベース）
- **暗号化方式**
  - 公開鍵暗号（RSA、Elliptic Curve）
  - 共通鍵暗号（AES）

リフレッシュトークンはアクセストークンを再発行する際に使われるトークンである。

リフレッシュトークン

- アクセストークンの再発行のために使われるトークン。
- トークンリフレッシュのリクエストをする際には、トークンエンドポイントにリクエストを行う。その際には”grant_type”を”refresh_token”パラメータに指定する必要がある。
- クライアントは自身のクライアントIDと紐づいた認証方式で、トークンエンドポイントに対して自身を認証しなければならない。
- トークンリフレッシュのレスポンスとして、新しく発行されたアクセストークンとリフレッシュトークンが返される。

リフレッシュリクエスト

```
POST /token HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded
client_id=s6BhdRkqt3
&client_secret=some_secret12345
&grant_type=refresh_token
&refresh_token=8xLOxBtZp8
&scope=openid%20profile
```

リフレッシュレスポンス

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache
{
 "access_token": "TlBN45jURg"、
 "token_type": "Bearer"、
 "refresh_token": "9yNOxJtZa5"、
 "expires_in": 3600
}
```

Implementation ConsiderationではOPとRPの両者により利用される機能について定義されている。

全てのOPに実装が求められる機能

- IDトークンの署名でRSA-SHA256をサポートしなければならない。（OPがIDトークンをTokenエンドポイントのみから返却し、IDトークン署名アルゴリズムをnoneを要求した状態でのみクライアント登録を許可する場合は、その限りでない。）
- 認可コードフローの認可要求でpromptパラメータをサポートしなければならない。これには none や login などが規定するユーザインタラクションの実装を含む。
- OPはdisplayパラメータをサポートしなければならない。
- ui_locales と claims_locales パラメータによって、ユーザインターフェースと Claim 用の推奨の言語や文字種のリクエストをサポートしなければならない 。
- 要求があれば、エンドユーザが認証を行った時刻を返却しなければならない。
- 最大認証期間をサポートしなければならない。
- 特定の認証コンテキストクラスリファレンス(ACR)の値のリクエストをサポートしなければならない。

Implementation ConsiderationではOPとRPの両者により利用される機能について定義されている。

動的クライアント登録をサポートしている動的OPに実装が求められる機能

- Response Typeは、id_token、またself-Issued OPではないOPは、codeとit_token tokenをサポートしなければならない。
- OpenID Connect Discovery 1.0で定義されているように Discovery 機能をサポートしなければならない。
- OpenID Connect Dynamic Client Registrationで定義されているように, 動的クライアント登録をサポートしなければならない。
- アクセストークンを発行する動的OPは、UserInfoエンドポイントをサポートしなければならない。
- OP は自身の公開鍵を素の JWK 鍵として公開しなければならない。
- request_uriパラメータで提供されたリクエストURIから取得するリクエストObject 値を使ってリクエストを送信できるようにしなければならない。

Discovery登録

- OpenID Connect 導入にあたっては、事前に設定されたOPおよびRPのセットが利用可能なこともある。そのような場合には、Identity やサービス、動的クライアント登録についての情報に関する Dynamic Discovery をサポートする必要は無いかもしれない。
- しかし、事前登録のないRPとOP間の予期しないやりとりをサポートすることを選択したならば、OpenID Connect Dynamic Client Registration 1.0の仕様で定義されているファシリティを実装するべきである。

Implementation ConsiderationではOPとRPの両者により利用される機能について定義されている。

RPに実装が求められる機能

- RPがバックエンドサーバのようなConfidentialクライアントは認可コードフローを、ネイティブアプリやフロントエンドになる場合はインプリシットフローが適切であり、それぞれの認証フローで必須となる機能は実装が必要となるが、任意の機能はサポートする必要が無い。

OpenID Connectでは、OAuth2.0のSecurity ConsiderationsとISO/IEC 29115を参照し、実装者が考慮する必要がある脅威のリストを提供している。

Security Considerationの項目一覧

項番	脅威	対策・考慮事項
1	適切な処置が講じられないとき、 リクエストは攻撃者に漏洩され、 セキュリティとプライバシーの脅威をもたらすかもしれない。	RFC6819に加え、request パラメータ (request の内容は適切な鍵と暗号により暗号化されたJWT) もしくは request_uri パラメータを利用することによりエンドツーエンドでリクエストの機密性を提供する方法を提供する。
2	悪意のあるサーバは、様々な方法を用いて正規のサーバに成りすますかもしれない。	RFC6819に加え、Signed JWTとEncrypted JWTの両方でサーバ認証をする方法を提供する。
3	攻撃者は偽のトークンを生成もしくは既存の解析可能なトークンの内容(クレームや署名など) を修正し、 RPにクライアントへの不適切なアクセスを引き起こすかもしれない。	このリスク軽減のために、OPがトークンにデジタル署名を付ける、トークンをTLSのような保護されたチャンネルで送信するの2つの方法がある。
4	-	アクセストークンを認可されていない対象にさらされることはあってはならない。
5	サーバーのレスポンスは、 認証データとセンシティブなクライアント情報を含むクレームを含む可能性がある。	認可コードフローを使う。 レスポンスはTLSで保護されたチャンネル上で送信し、クライアントは、クライアントIDとクライアントシークレットで認証される。
6	適切なメカニズムが実施されていないならば、 レスポンスはサーバにより拒否されるかもしれない。	この脅威を軽減するため、レスポンスは否認防止のための鍵を用いてサーバにより署名されてもよい。 クライアントはそれが正規のサーバから発行されたものであり、完全であることを確認するためにデジタル署名を検証すべきである。
7	脆弱または悪意のあるクライアントがリクエストを誤った対象に送ることがありえるため、 bearer token のみを利用して認証されたクライアントはいかなるトランザクションでも拒否されることができる。	この脅威を軽減するため、サーバはクライアントにより否認防止のための鍵を用いてデジタル署名されたリクエストを要求してもよい。 サーバはそれが正規のクライアントから発行されたものであり、完全であることを確認するためにデジタル署名を検証すべき。

引用元：http://openid-foundation-japan.github.io/openid-connect-core-1_0.ja.html

OpenID Connectでは、OAuth2.0のSecurity ConsiderationsとISO/IEC 29115を参照し、実装者が考慮する必要がある脅威のリストを提供している。

Security Considerationの項目一覧（続き）

項番	脅威	対策・考慮事項
8	攻撃者は、あるリソースのために生成されたアクセストークンを別のリソースへのアクセスするために利用する。	アクセストークンは audience とスコープに対して制限をかけるべき。
9	攻撃者は、対象のリソースのために既に一度利用された認可コードのようなワンタイムなトークンの再利用を試みる。	トークンはタイムスタンプを含み、有効期限を短くすべき。 トークンが現時点で有効であることを確実にするため、 タイムスタンプと有効期限の値をチェックする。
10	User Agentがマルウェアに感染している場合は、[RFC6819] Section 4.4.1.1 にある攻撃パターンに加え、認可コードが TLS セッションの終端となる User Agent 上でも盗聴される可能性がある。	クライアントと認証やレスポンス暗号化を用いると、 この盗聴パターンは無効化することができる。
11	不正なユーザがセッション中からトークンを取り出し、別のセッション中で利用するトークンを置換するタイプの攻撃がある。交換されるトークンには認可コードも含まれる。 また不正なユーザが、認可エンドポイントとクライアントの間、 もしくはトークンエンドポイントとクライアントの間の通信に介入し、認可コードを置換したりメッセージの順序を入れ替えたりすることで、 特権ユーザになりすます可能性もある。	OAuth2.0 インプリシットフローでは、このリスクを考慮した設計になっていないが、OpenID Connect では、 ID トークンを利用することでこのリスクを回避できる。 トークンを含んだレスポンスとリクエストが TLS で保護されている限り、 パケットの順序入れ替えは検知および防御可能であろう。
12	タイミングアタックにより、攻撃者は復号処理や署名検証の成功/失敗にかかる処理時間の違いを通じて、必定以上に多くの情報を得ることができる.	実装者はエラーを検知した際すぐに検証プロセスを終了するのではなく、 他のオクテットに対する処理を終えるまで処理を継続し、この攻撃を防ぐべきである。

引用元：http://openid-foundation-japan.github.io/openid-connect-core-1_0.ja.html

OpenID Connectでは、OAuth2.0のSecurity ConsiderationsとISO/IEC 29115を参照し、実装者が考慮する必要がある脅威のリストを提供している。

Security Considerationの項目一覧（続き）

項番	脅威	対策・考慮事項
13	暗号、 署名、 完全性検証の手法に応じて、 暗号関連の様々な攻撃が考えられる	実装者は JWT [JWT] の Security ConsiderationsおよびJWT が参照する各脆弱性を防ぐための仕様群に従うこと。
14	暗号文に対する署名は、 多くの法域 (jurisdiction: 法律用語、 ある法律が適用される地域のこと) で有効と認められない。	完全性と非否認性を確保するため署名を利用する際には、 平文のJSONクレームに対して署名をすることを要求する。 署名と暗号化がともに必要となる場合、 署名対象のクレームを含んだ JWS に対して暗号化を行うこと。
15	-	ディスカバリーで返される Issuerは、IDトークンに含まれる issと完全一致しなければならない。
16	-	インプリシットフローでは、User Agentがマルウェアに感染していたり不正な操作者のコントロール下にある場合は、 TLSセッションの終端となる User Agent 上でアクセストークンを読み取られることもありうる。
17	-	各実装はTLSをサポートしなければならない。

引用元：http://openid-foundation-japan.github.io/openid-connect-core-1_0.ja.html

OpenID Connectでは、OAuth2.0のSecurity ConsiderationsとISO/IEC 29115を参照し、実装者が考慮する必要がある脅威のリストを提供している。

Security Considerationの項目一覧（続き）

項番	脅威	対策・考慮事項
18	-	認可サーバはアクセストークンを無効化できないかもしれない。従って、アクセストークンの有効期間は一度の利用に十分な期間、もしくは非常に短い期間に限定するべきである。 UserInfoエンドポイントや他の保護されたリソースに対する継続したアクセスが必要な場合は、有効期限の短いアクセストークンとリフレッシュトークンを活用するよい。 認可サーバは認証中にユーザに対して長期間の認可を与えようとしていることを明示すべきである。　認可サーバはエンドユーザに対して、特定のクライアントに発行されたアクセストークンおよびリフレッシュトークンを無効化する手段を提供すべきである 。
19	-	共通鍵暗号による署名および暗号化を行うためには、クライアントシークレットが十分なエントロピーを持った暗号論的に強固な鍵とする必要がある。 クライアントシークレットは利用するアルゴリズムが利用する MAC 鍵に必要な最低限のオクテット長を持つ必要がある。例えば HS256 を利用する際は、 クライアントシークレットは最低でも32オクテットが必要である（なお、 多くのの場合クライアントシークレットはアルファベットに限定されるため、 明らかにそれ以上のオクテットが必要となるであろう。
20	-	場合によっては、クライアントはリクエストパラメータの改竄を防止するため、 リクエストに署名を行う必要があるかもしれない. 例えば max_age と acr_valuesなどは、署名付きリクエストを用いた方がより正確に認証処理についての保証として有用になるであろう。
21	-	エンドユーザに関するセンシティブな情報の漏洩を防ぐため、OPへのリクエストを暗号化をおこなう。

引用元：http://openid-foundation-japan.github.io/openid-connect-core-1_0.ja.html

OpenID Connect Coreでは、個人情報・アクセス監視・クライアントの相関関係・オフラインアクセスに関しての、プライバシー保護について記載されている。

プライバシー保護に関する記載 (Privacy Considerations)

- 個人情報
  - UserInfo レスポンスには、個人情報が含まれる。そのため、特定の目的のためにエンドユーザーの情報を公開する際に取得する同意は、承認時または承認時よりも前に取得しなければならない。
  - クライアントには必要なUserInfoデータのみを保存し、受信したデータと利用目的を関連付けるべきである。
- データアクセス監視
  - リソースサーバはエンドユーザの UserInfoへのアクセスログを取得し、誰がデータにアクセスしたかを監視できるようにすべきである。
- 相関関係
  - クライアント間の相関関係からエンドユーザーを保護するために、sub値はpairwise識別子の使用を考慮すべき。
- オフラインアクセス
  - オフライン・アクセスはユーザーが不在の場合にクレームへのアクセスを可能にし、プライバシーリスクをもたらす。リソースへのオフラインアクセスについては明示的な同意を得ることが賢明である。

2. OpenID Connect Client Initiated Backchannel Authentication Flow (CIBA)

CIBAはMODRNA WGにおいて策定されている仕様で、2020年1月に公開されているドラフトの第3版が最新版である。

目次（青字でタイトルを記載している章は、詳しく後述）

章番号	タイトル	概要
1	**Introduction**	CIBAに関するイントロダクション。認証したいユーザの有効な識別子を取得できるRPは、消費デバイス(Consumption Device: CD)からエンドユーザとの対話を行うことなく、ユーザを認証するための対話フローを開始することができることを述べている。
2	**Teminology**	CIBAで新しく使う用語の説明に関して記述されている。
3	**Overview**	CIBAの仕様の概要説明。CIBAでは、クライアントがOPから各種トークンを受け取る際には、3つのモードがあることなどが記載されている。
4	Registration and Discovery Metadata	CIBAのクライアント登録とdiscoveryに関してサポートしなければならない仕様に関して記載がされている。
5	**Poll, Ping and Push Modes**	クライアントが各種トークンを受け取る際の3つのシーケンスに関して説明している。
6	**Example Use Cases**	CIBAを使う際のユースケースが例示されている。
7	**Backchannel Authentication Endpoint**	エンドユーザの帯域外認証を開始するために使われる、バックチャネル認証エンドポイントの仕様について記載されている。
8	OpenID Provider Obtains End-User Consent/Authorization	OPがユーザを識別した後、ユーザの認証デバイスで対話が開始し、承認の決定を得ることが述べられている。

CIBAはMODRNA WGにおいて策定されている仕様で、2020年1月に公開されているドラフトの第3版が最新版である。

目次（続き）

章番号	タイトル	概要
9	**Client Notification Endpoint**	クライアント通知エンドポイントはエンドユーザー認証が成功または失敗した後OP が呼び出すエンドポイントであり、その仕様が記載されている。
10	**Getting the Authentication Result**	クライアントがOPから各種トークンを受け取る際に考慮すべき事項について記載されている。
11	Token Error Response	トークンリクエストが無効・不正だった場合に通知するエラーメッセージに関して記載されている。
12	Push Error Payload	プッシュトークン配信モードを使用するようにクライアントが設定されている場合、クライアント通知エンドポイントでエラーペイロードを受信できることが記載されている。
13	Authentication Error Response	認証エラーレスポンスのエラーコードに関して規定されている。
14	**Security Considerations**	CIBAをサポートするにあたってセキュリティで考慮すべき事項についてまとめられている。
15	Privacy Considerations	CIBAをサポートするにあたって個人情報保護の観点から考慮すべき事項について述べられている。
16	IANA Considerations	本書に定義したパラメータを、別の仕様書に登録する際に要求される事項について記載している。

CIBAは新しく定義された認証フローであり、RPはエンドユーザとのやりとりを行うことなく、認証フローを開始することができる。

認可コードフローとCIBAで定義されているフロー

- OpenID Connect Coreで定義される認可コードフローとCIBAで定義されているフローの違いは下記の通り。

用語	認可コードフロー	CIBAで定義されているフロー
認可要求 (認証要求)	ユーザーのUAを介したリクエスト。(リダイレクト)	UAを介さず直接リクエスト。 RPはユーザーを特定するヒントを送る必要がある。
（認証）	一般的にはUAに表示されるログイン画面で認証。	OPでRPが送付したユーザーを識別する情報を検証。 必要があれば認証デバイスでも認証。
（同意）	一般的にはUAに表示される許諾画面で同意。	ユーザーの認証デバイス(Authentication Device: AD)上で同意。
トークン要求	同意後にUAを介して返却された認可コードをトークンに引き換える。	3種類のモードで同意済みかどうかを確認し、トークンを取得する。

引用：https://www.secure-sketch.com/blog/openid-connect-ciba-implementation

CIBAでは、OpenID Connect CoreやOAuth 2.0で定義された用語に加えて、消費デバイスや認証デバイスといった新たな用語が定義されている。

CIBAにおける用語

- CIBAでは、OpenID Connect Coreで定義されているOPとRPの用語と、OAuth2.0で定義されているクライアントの用語を使用している。
- CIBAの仕様書では上記に加え、「消費デバイス」と「認証デバイス」の２つの用語を新しく定義している。

用語	用語の規定元	概要
OpenID Provider(OP)	OpenID Connect Core1.0	ユーザの認証およびIDトークン、アクセストークンの発行機能を有するサーバで、ユーザ認証後にRPからの要求に対してIDトークンとアクセストークンの発行を行う。
Relying Party(RP)	OpenID Connect Core1.0	OPから認証結果を受け取り、OPに対してアクセストークンとIDトークンの発行を要求し、Access Tokenを使ってUserInfoエンドポイントに対して要求を行いアイデンティティ情報を入手する。
クライアント	OAuth 2.0	保護されたリソースの所持者から認可を得て、リソースオーナーの代理として保護されたリソースに対するリクエストを行うアプリケーション。
消費デバイス（CD）	OpenID Connect Client Initiated Backchannel Authentication Flow - Core 1.0	消費デバイスは、ユーザの消費サービスを支援するデバイス。CIBAのユースケースでは、ユーザは必ずしも消費デバイスを制御しているわけではない。
認証デバイス（AD）	OpenID Connect Client Initiated Backchannel Authentication Flow - Core 1.0	認証デバイス（Authentication Device） ユーザがリクエストを認証するデバイス。多くの場合はスマートフォンを想定している。

CIBAではクライアントが直接エンドユーザとやりとりすることなく認証フローを開始することが可能。その際の各種トークン受取方式として3つのモードが定義されている。

CIBAの特徴

- CIBAは、クライアントが直接エンドユーザとやりとりすることなく、認証フローを開始することが可能な方式となっている。
- CIBAフローにおいて、クライアントが認可サーバからID トークン、アクセストークンおよびリフレッシュトークン(オプション)を取得する方式(モード)としてPoll、Ping、Pushの3つが定義されている。

モード	概要
Poll	Pollモードで設定されている場合、クライアントはトークンエンドポイントをpollして、トークンレスポンスを取得する。
Ping	Ping モードで設定されている場合、OPは、バックチャネル認証エンドポイントから返された一意の識別子で、クライアントが事前に登録したコールバックURI にリクエストを送信する。通知を受信すると、クライアントはトークンを取得するためにトークンエンドポイントにリクエストを行う。
Push	プッシュモードで設定すると、OPはクライアントが事前に登録したコールバック URI にトークン付きのリクエストを送信する。

クライアントは各種トークンの受信の処理において3つのモードを選択することが可能。

CIBAにおけるトークン受信モード

- クライアントがトークンを取得する方法として、Poll、Ping、Pushの３つの方法が定義されている。

モード	Poll	Ping	Push
概要	RPはトークンエンドポイントに対してポーリング(定期的にリクエスト)を行う。 AD上でユーザ同意が完了した後にリクエストを行うと、トークンが返却される。	AD上でユーザの同意が完了すると、OPはRPに対して、完了通知を送る。 通知を受け取ったRPはトークンリクエストを行う。	AD上でユーザの同意が完了すると、OPはRPに対してトークンを返却する。
イメージ			

引用：https://www.secure-sketch.com/blog/openid-connect-ciba-implementation

CIBAの仕様内において、CIBAのユースケースとして３点の具体例が紹介されている。

CIBAのユースケース

① コールセンター業務における、コールセンターの代理人が発信者の認証を通しての発信者の情報確認。

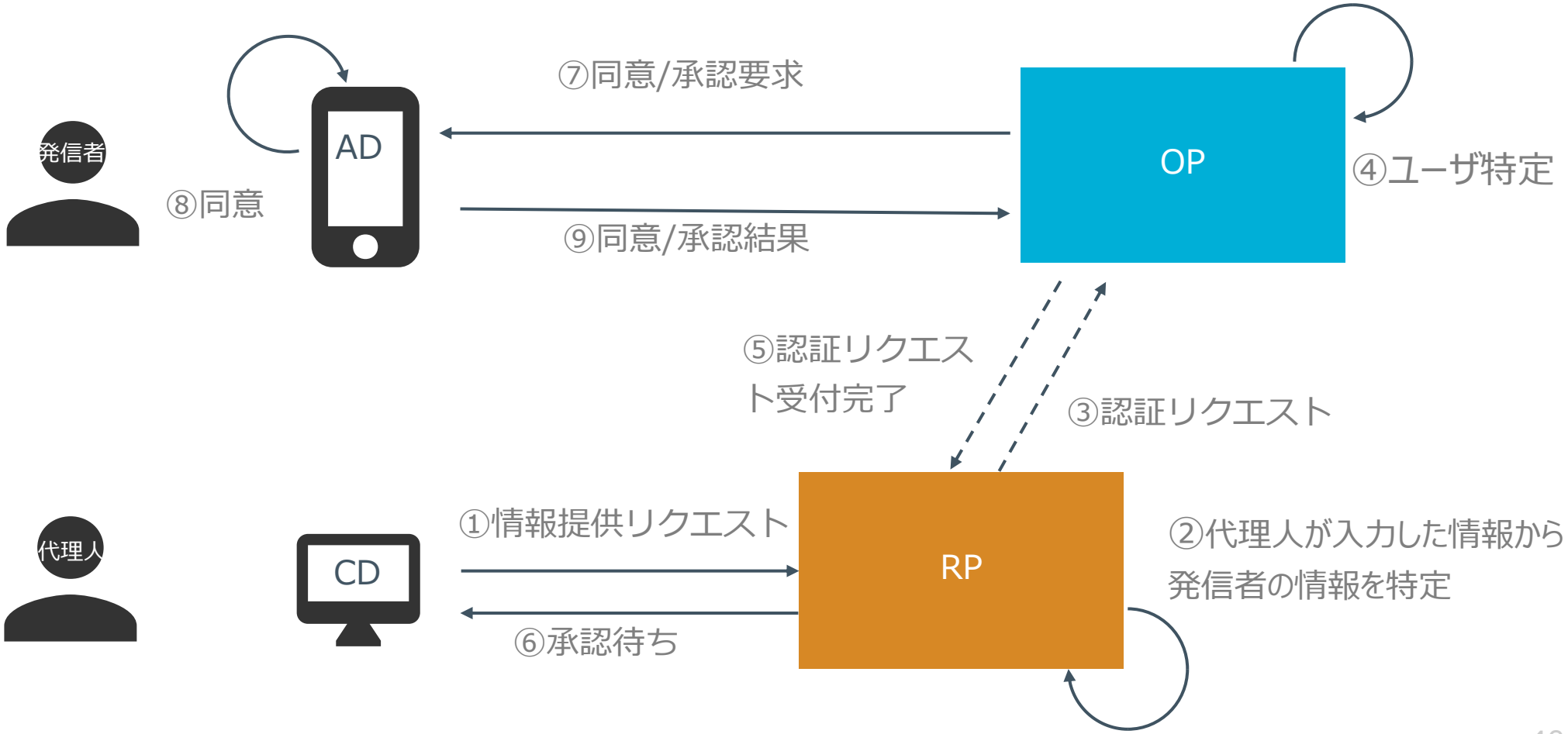
② 銀行支店での業務における、銀行の窓口の担当者による対面での顧客認証。

③ 店頭販売における、スマートフォンを利用した店頭端末での支払いの認証。

- 以降では、①のコールセンター業務におけるCIBA導入例について詳細を記載する。

コールセンターにおいて、CIBAの活用により発信者の情報を参照する端末と発信者の認証を行う端末を分離することが可能となる。

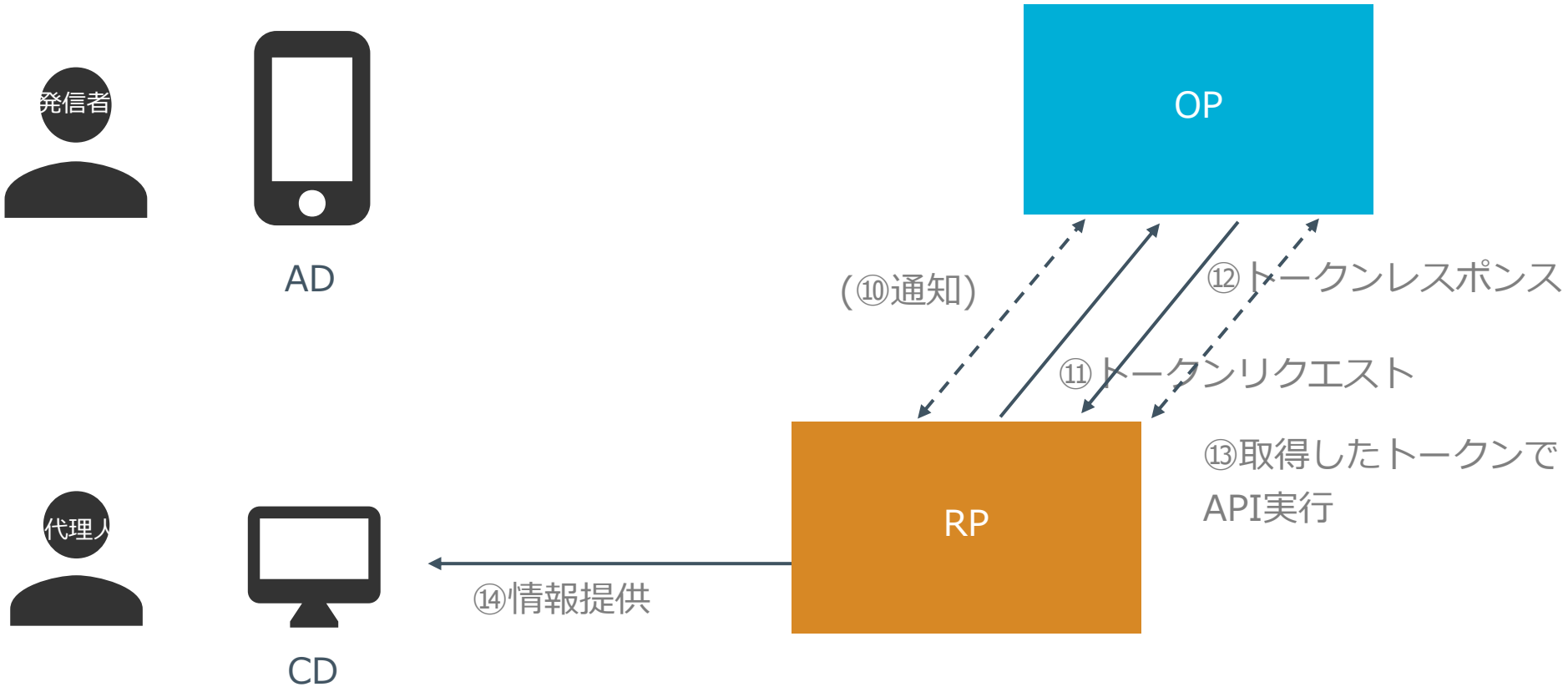
- 「①コールセンター業務における、コールセンターの代理人が発信者の認証を通しての発信者の情報確認」におけるフロー



引用：https://www.secure-sketch.com/blog/openid-connect-ciba-implementation

コールセンターにおいて、CIBAの活用により発信者の情報を参照する端末と発信者の認証を行う端末を分離することが可能となる。

- 「①コールセンター業務における、コールセンターの代理人が発信者の認証を通しての発信者の情報確認」におけるフロー

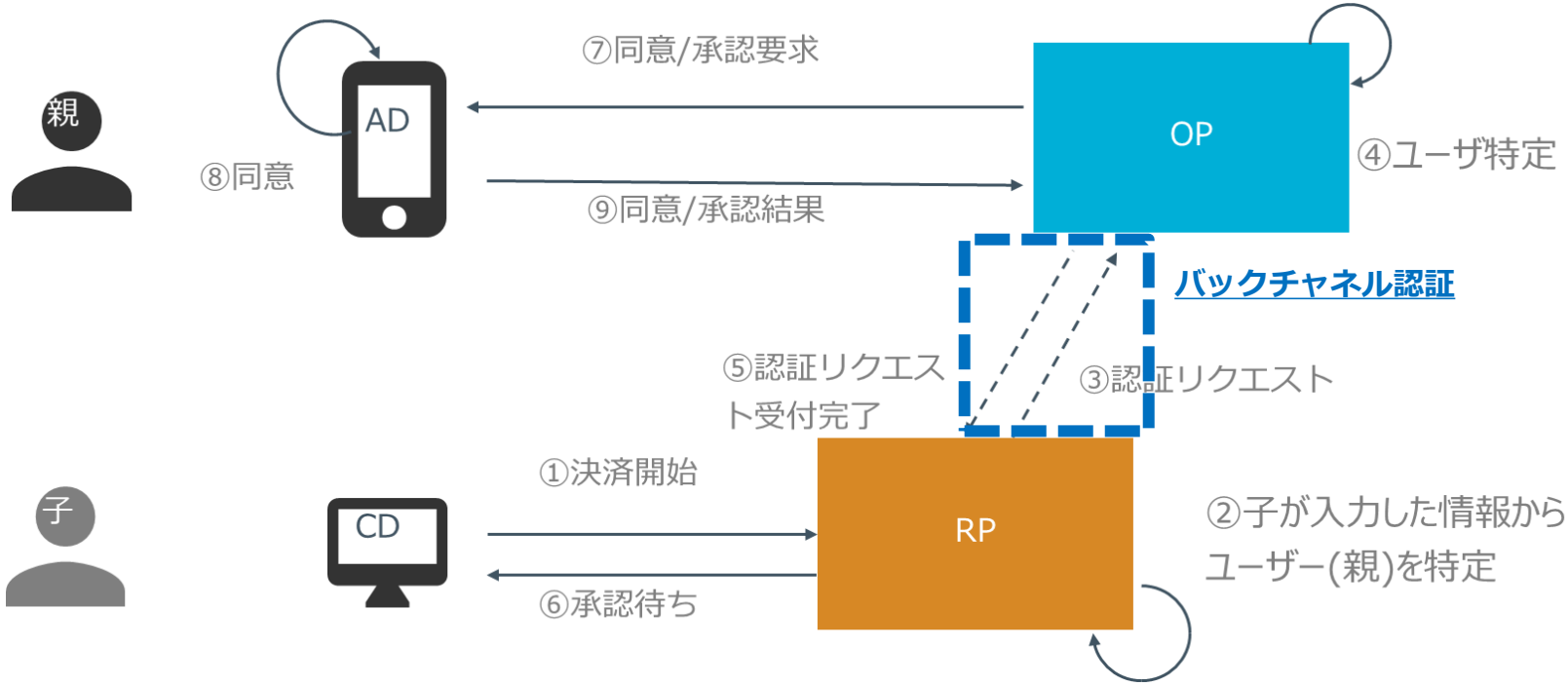


引用：https://www.secure-sketch.com/blog/openid-connect-ciba-implementation

バックチャネル認証は、ユーザーのブラウザを経由せずにクライアントからOPに直接要求される認証リクエストである。

バックチャネル認証

- クライアントは OPに認証リクエストを送り、クライアントIDに登録された認証方法を使ってバックチャネル認証エンドポイントに認証を実施。
- OPは対象の利用者であることを識別する値としてその Issuer Identifier、トークンエンドポイントURL、または バックチャネル認証エンドポイントURLを受け入れなければならない。



引用：https://www.secure-sketch.com/blog/openid-connect-ciba-implementation

バックチャネル認証のリクエスト例

バックチャネル認証

- クライアントはクライアントIDに登録された認証方法を使いバックチャネル認証エンドポイントに認証しなければならない。
- 下記に、バックチャネル認証エンドポイントへのリクエスト例とその各パラメータの説明を示す。

パラメータ	必須	説明
scope	Required	ユーザ情報取得のためのアクセス要求のスコープ。CIBAでは、openidを指定する必要がある。
client_notification_token	Required	クライアントが Ping またはPushモードを使用する場合、必須。クライアントから提供されるベアラートークンで、OPがクライアントへのコールバックリクエストを認証するために使用する。
login_hint_token	Optional	認証が要求されるエンドユーザーを識別する情報を含むトークン。
client_assertion_type	条件付き	JWT Assertionでのクライアント認証に使われるパラメータで、「urn:ietf:params:oauth:client-assertion-type:jwt-bearer」にする必要がある。（Mutual TLS クライアント認証も可）
client_assertion	条件付き	JWT Assertionでのクライアント認証に使われるパラメータで、クライアント認証情報を含むJWTを設定する必要がある。

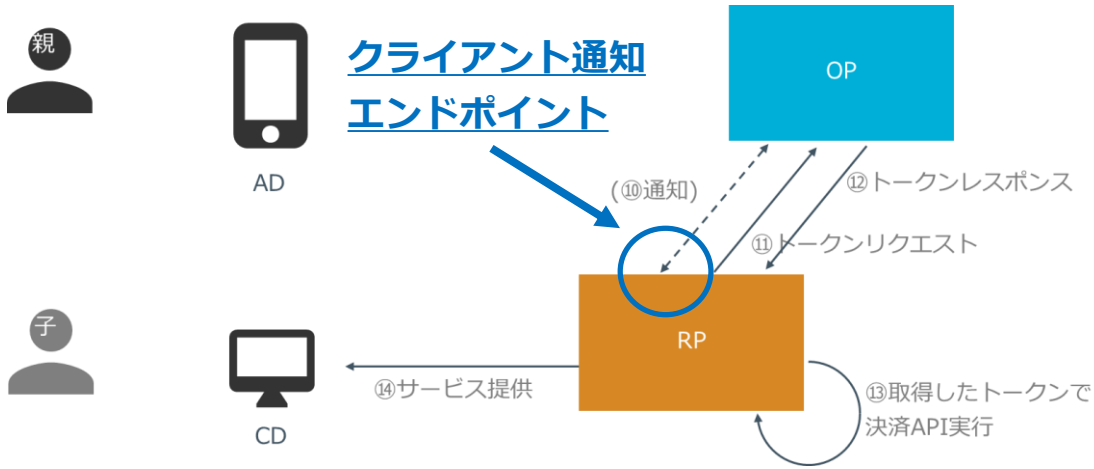
```
POST /bc-authorize HTTP/1.1
  Host: server.example.com
  Content-Type: application/x-www-form-urlencoded

  scope=openid%20email%20example-scope&
  client_notification_token=8d67dc78-7faa-4d41-aabd-67707b374255&
  binding_message=W4SCT&
  login_hint_token=eyJraWQiOiJsdGFjZXNidyIsImFsZyI6IkVTMjU2In0.eyJ
  zdWJfaWQiOnsic3ViamVjdF90eXBlIjoicGhvbmUiLCJwaG9uZSI6IisxMzMwMjg
  xODAwNCJ9fQ.Kk8jcUbHjJAQkRSHyDuFQr3NMEOSJEZc85VfER74tX6J9CuUllr8
  9WKUHUR7MA0-mWlptMRRhdgW1ZDt7g1uwQ&
  client_assertion_type=urn%3Aietf%3Aparams%3Aoauth%3A
  client-assertion-type%3Ajwt-bearer&
  client_assertion=eyJraWQiOiJsdGFjZXNidyIsImFsZyI6IkVTMjU2In0.eyJ
  pc3MiOiJzNkJoZFJrcXQzIiwic3ViIjoiczZCaGRSa3F0MyIsImF1ZCI6Imh0dHB
  zOi8vc2VydmVyLmV4YW1wbGUuY29tIiwianRpIjoiYmRjLVhzX3NmLTNZTW80RlN
  6SUoyUSIsImlhdCI6MTUzNzgxOTQ4NiwiZXhwIjoxNTM3ODE5Nzc3fQ.Ybr8mg_3
  E2OptOSsA8rnelYO_y1L-yFaF_j1iemM3ntB61_GN3APe5cl_-5a6cvGlP154XAK
  7fL-GaZSdnd9kg
```

クライアント通知エンドポイントは、エンドユーザが認証に成功または失敗した後、OPが呼び出すエンドポイントである。

クライアント通知エンドポイントの定義

- クライアントがPingモードで設定されている場合、エンドポイントは、トークンエンドポイントから認証結果を取得する準備ができたという通知をOPから受信する。
- クライアントがPushモードで設定されている場合、エンドポイントは認証結果(ID トークン、アクセ ストークン、オプションでリフレッシュトークン、またはユーザーが認証を許可しなかった場合はエラー)を受信する。
- クライアント通知エンドポイントへのリクエストは、クライアントによって作成されたベアラートークンを使用して認証されなければならず、パラメータclient_notification_tokenの値として認証リクエストでOPに送信される。

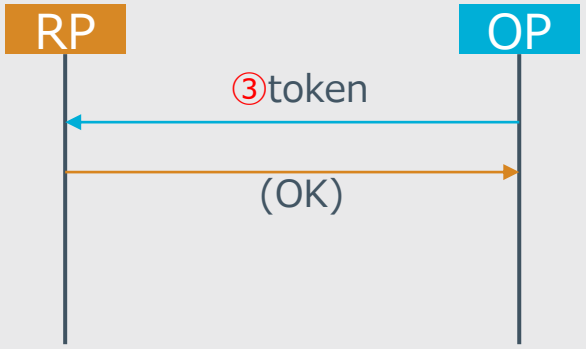


引用：https://www.secure-sketch.com/blog/openid-connect-ciba-implementation

クライアントが各種トークンを受信する方式(モード)として3つのモードが定義されている。

概要とイメージ

- クライアントがトークンを取得する方法は、Poll、Ping、Pushの３つの方法がある。
- **下の表中のイメージ内に記載の①、②、③のメッセージについてそれぞれ詳細を後述する。**

モード	Poll	Ping	Push
概要	RPはトークンエンドポイントに対してポーリング(定期的にリクエスト)を行う AD上でユーザ同意が完了した後にリクエストを行うと、トークンが返却される	AD上でユーザの同意が完了すると、OPはRPに対して、完了通知を送る 通知を受け取ったRPはトークンリクエストを行う	AD上でユーザの同意が完了すると、OPはRPに対してトークンを返却する
イメージ			

引用：https://www.secure-sketch.com/blog/openid-connect-ciba-implementation

メッセージ①：クライアントがPollまたはPingモードを使用する場合、トークンエンドポイントに対してリクエストを行い、トークンの取得を行う。

トークンエンドポイントへのリクエスト

パラメータ	必須	説明
grant_type	Required	CIBAのフローであるため、「urn:openid:params:grant-type:ciba」と設定する必要がある。
auth_req_id	Required	クライアントが行った認証要求を識別するための一意の識別子。
client_assertion_type	条件付き	JWT Assertionでのクライアント認証に使われるパラメータで、「urn:ietf:params:oauth:client-assertion-type:jwt-bearer」を指定する必要がある。
client_assertion	条件付き	JWT Assertionでのクライアント認証に使われるパラメータで、クライアント認証情報を含むJWTを設定する必要がある。

```
POST /token HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded

grant_type=urn%3Aopenid%3Aparams%3Agrant-type%3Aciba&
auth_req_id=1c266114-a1be-4252-8ad1-04986c5b9ac1&
client_assertion_type=urn%3Aietf%3Aparams%3Aoauth%3A
client-assertion-type%3Ajwt-bearer&
client_assertion=eyJraWQiOiJsdGFjZXNidyIsImFsZyI6IkVTMjU2In0.ey
Jpc3MiOiJzNkJoZFJrcXQzIiwic3ViIjoiczZCaGRSa3F0MyIsImF1ZCI6Imh0d
HBzOi8vc2VydmVyLmV4YW1wbGUuY29tL3Rva2VuIiwianRpIjoiLV9wMTZqNkhj
aVhvMzE3aHZaMzEyYyIsImlhdCI6MTUzNzgxOTQ5MSwiZXhwIjoxNTM3ODE5Nzg
yfQ.BjaEoqZb-81gE5zz4UYwNpC3QVSeX5XhH176vg35zjkbq3Zmv_UpHB2ZugR
Va344WchTQVpaSSShLbvha4yziA
```

メッセージ②：クライアントがPingモードの場合、OPでエンドユーザが認証に成功・失敗した後、クライアント通知エンドポイントにリクエストを送信する。

Pingモード時の認証結果通知のリクエスト

- Authorizationヘッダーにベアラートークンとしてclient_notification_tokenをセットし、リクエストの本文にはauth_req_idを含めて送信する。

```
POST /cb HTTP/1.1
Host: client.example.com
Authorization: Bearer 8d67dc78-7faa-4d41-aabd-67707b374255
Content-Type: application/json

{
 "auth_req_id": "1c266114-a1be-4252-8ad1-04986c5b9ac1"
}
```

メッセージ③：クライアントがPushモードで登録されている場合、OPはクライアント通知エンドポイントにIDトークンやアクセストークンなどを含むペイロードを送信する。

Pushモード時の認証結果通知

- IDトークン、アクセストークンおよびauth_req_idを紐づけるために、アクセストークンとauth_req_idのハッシュ値をat_hashとurn:openid:params:jwt:claim:auth_req_idクレームを用いて下記のようにIDトークン内に含める。
- リフレッシュトークンを送信する場合、urn:openid:params:jwt:claim:rt_hashクレームを使用してそのハッシュ値もIDトークンに追加しなければならない。
- Pushモード時にOPからクライアント通知エンドポイントに送信される認証結果とIDトークンのペイロード部の例を以下に示す。

```
 POST /cb HTTP/1.1
 Host: client.example.com
 Authorization: Bearer 8d67dc78-7faa-4d41-aabd-67707b374255
 Content-Type: application/json

{
  "auth_req_id": "1c266114-a1be-4252-8ad1-04986c5b9ac1",
  "access_token": "G5kXH2wHvUra0sHlDy1iTkDJgsgUO1bN",
  "token_type": "Bearer",
  "refresh_token": "4bwc0ESC_IAhflf-ACC_vjD_ltc11ne-
8gFPfA2Kx16",
  "expires_in": 120,
  "id_token":
"eyJhbGciOiJSUzI1NiIsImtpZCI6IjE2NzcyNiJ9.eyJpc3.........”
}
```

Pushモード時のクライアント通知エンドポイントに送信される認証結果

```
{
  "iss": "https://server.example.com",
  "sub": "248289761001",
  "aud": "s6BhdRkqt3",
  "email": "janedoe@example.com",
  "exp": 1537819803,
  "iat": 1537819503,
  "at_hash": "Wt0kVFXMacqvnHeyU0001w",
  "urn:openid:params:jwt:claim:rt_hash":
"sHahCuSpXCRg5mkDDvvr4w",
  "urn:openid:params:jwt:claim:auth_req_id":
        "1c266114-a1be-4252-8ad1-04986c5b9ac1“
}
```

IDトークンのペイロード部

CIBAを利用する際にセキュリティ面での考慮事項についてSecurity Considerations としてまとめられている。

項番	脅威	対策・考慮事項
1	インジェクション攻撃、不正なクライアントによるユーザー識別子の収集	login_hint_tokenは、発行者によってデジタル署名されるべき。これにより、データの真正性を保証し、インジェクション攻撃の脅威を軽減することができる。また、OPは署名により、ヒントの送信者を認証することができ、不正なクライアントによるユーザIDの収集を防ぐことができる。
2	OP は認証結果を間違ったクライアントに認証結果を送る恐れがある。	OPは登録時に設定した「backchannel_client_notification_endpoint」がクライアントの管理権限下にあることを確認すべきである
3	-	ID_Token_hintは、標準的なJWT処理ルールを使用して検証することができない。したがって、OPは、ある程度の期間、有効期限が過ぎたid_token_hintを受け入れるべきである。 OPは、トークンの発行者であることと、id_token_hintを提示したクライアントがaudienceクレームに記載されていることを確認しなければならない。 OPは、改ざん検知のため署名を検証しなければならない。しかし、鍵のローテーションのため、OPは必ずしもトークンの署名に使用された鍵にアクセスできるとは限らないことに注意すること。 このような制約から、実装者は署名の検証を一切行わず、pairwise subject identifiersをヒントとする ID トークンのみを受け入れることを検討してもよい。そうすれば、OPは、バックチャネル認証エンドポイントで認証されたクライアントがペアワイズのpairwise subject identifiersを発行されたことを検証することができる。

CIBAを利用する際にセキュリティ面での考慮事項についてSecurity Considerations としてまとめられている。

項番	脅威	対策・考慮事項
4	-	CIBAのPushモードでは、トークンはクライアント通知エンドポイントで直接クライアントに送信されるため、このエンドポイントと OPへの登録のために適切なセキュリティ管理が行われていることを保証しなければならない。 CIBA は、Push モードを使用する場合、アクセストークンのハッシュ、リフレッシュトークンのハッシュ値、および auth_req_idがIDトークンに含まれていることを要求する。これにより、クライアントは Push コールバックの値が改ざんされていないことを確認することができる。
5	-	シナリオによっては、RP がユーザーとのセッションのコンテキストに関するメタデータをOP に渡すのが適切な場合がある。例えば、RPは消費デバイスの地理情報を送信し、OPは認証デバイスの地理情報と照合することができる。 本仕様では、そのようなメタデータを送信することを要求するものではなく、また、そのようなメタデータが伝達される方法を定義するものでもない。

3. Health Relationship Trust Profile for OAuth 2.0

Health Relationship Trust Profile for OAuth 2.0 (HEART)は、 HEART WGにて策定されているプロファイルであり2020年1月にドラフトの第3版が公開されている。

- 目次は下記の通り（青字でタイトルを記載している章は、詳しく後述する）

章番号	タイトル	概要
1	**Introduction**	HEARTに関するイントロダクション。HEARTはOAuth2.0のウェブ認証フレームワークを、RESTful APIの安全性の確保を目指していることと、実装の容易さと幅広いユースケースに適したセキュリティ対策実装を目指して仕様の策定が行われていることが述べられている。
2	**Client Profiles**	OAuthグラントタイプに基づいて、様々なタイプのクライアントアプリケーションのプロファイルがまとめられている。
3	**Authorization Server Profile**	RFC6749のSecurity ConsiderationsとOAuth Threat Model Documentを参照した上で、認可サーバが準拠すべき事項について記載されている。
4	**Protected Resource Profile**	リソースサーバが対応すべき仕様に関してまとめられている。
5	**Advanced OAuth Security Options**	HEARTでは実装の容易さを維持しつつ、幅広いユースケースに適応したセキュリティレベルを提供するように仕様の策定が行われているが、実装の手間を犠牲にしてでもセキュリティを向上させたい場合の、セキュリティ対策に関してまとめられている。
6	**Security Considerations**	すべてのトランザクションは、TLSで保護されなければならないということと、 すべてのクライアントは、RFC6749のSecurity Considerationsのセクションにある適用可能な推奨事項と、OAuth 2.0 Threat Model and Security Considerationsのドキュメントにある推奨事項に準拠しなければならないということを述べている。

HEARTに関する主な用語

用語定義

用語	用語の規定元	概要
アクセストークン	OAuth 2.0	クライアントがリソースサーバにアクセスするために使われるトークン。
認可コード	OAuth 2.0	認可コードは、認可サーバーがクライアントとリソースオーナーの仲介となって発行する。 クライアントはソースオーナーを認可サーバーへリダイレクトさせ、リソースオーナーがリダイレクトして戻ってきた際に認可コードを取得する。
認可エンドポイント	OAuth 2.0	クライアントが認可サーバに認可リクエストを行う際に呼び出すエンドポイント。
認可グラント	OAuth 2.0	リソースオーナーによる (保護されたリソースへのアクセスを行うことに対する) 認可を示し, クライアントがアクセストークンを取得する際に用いられる。RFC6749では認可コード、インプリシット、リソースオーナーパスワードクレデンシャル、クライアントクレデンシャルの4つのグラントタイプが定義されている。
認可サーバ	OAuth 2.0	アクセストークンをクライアントに発行するサーバー
クライアント	OAuth 2.0	保護されたリソースの所持者から認可を得て、リソースオーナーの代理として保護されたリソースに対するリクエストを行うアプリケーション。
クライアント認証	OAuth 2.0	パスワードや公開鍵/秘密鍵のペア等を使って、認可サーバでクライアントの認証を行う。
クライアント識別子	OAuth 2.0	認可サーバに登録済みのクライアントに発行される識別子。
クライアントシークレット	OAuth 2.0	クライアントの秘密の値。ユーザ認証の際のID/PWのパスワードに相当。
グラントタイプ	OAuth 2.0	認可グラントの種別。

HEARTに関する主な用語

用語定義

用語	用語の規定元	概要
リフレッシュトークン	OAuth 2.0	現在のアクセストークンが無効化されたあるいは期限切れの際に新しいアクセストークンを取得するため、 使用されるクレデンシャル。
リソースオーナ	OAuth 2.0	保護されたリソースへのアクセスを許可できる実体で、所有権が人の場合はエンドユーザと呼ばれる。
リソースサーバ	OAuth 2.0	アクセストークンを使用して保護されたリソースに対する要求を受け入れて応答することができるサーバ。
レスポンスタイプ	OAuth 2.0	クライアントは”response_type”のパラメーターを利用して, 希望するグラントタイプを認可サーバに通知する。
トークンエンドポイント	OAuth 2.0	認可グラントもしくはリフレッシュトークンをアクセストークンと交換するために、クライアントに利用されるエンドポイント
クレームネーム	OpenID Connect Core 1.0 JSON Web Token (JWT)	クレーム表現の名前
クレーム値	OpenID Connect Core 1.0 JSON Web Token (JWT)	クレーム表現の値
JWT	OpenID Connect Core 1.0 JSON Web Token (JWT)	JWTは、JWSまたはJWE構造体にエンコードされた、JSONオブジェクトとしてクレームのセット。

第2章では、4つのクライアント種別が定義され、それぞれのクライアントが認可サーバおよび保護されたリソースにアクセスする際の考慮事項がまとめられている。

クライアントプロファイル

- 第2章では、 OAuthグラントタイプに基づいて、下記の各タイプのクライアントアプリケーションにおけるパターンが考慮されている。
  - フルクライアント（Webアプリケーションのバックエンドサーバなど）
  - ネイティブクライアント（デスクトップアプリケーションやスマートフォンアプリケーションなど）
  - フロントエンドクライアント（ブラウザ上のJavaScriptなど）
  - ダイレクトアクセスクライアント（信頼できるバックエンドサーバ同士で、リソースの参照を行うクライアント）
- 上記に加えて、それぞれのクライアントが下記に接続する際のプロファイルについてまとめられている。
  - 認可サーバ
  - トークンエンドポイント
  - 保護されたリソース
- さらに、下記のプロファイルについてもまとめられている。
  - クライアント登録
  - クライアントキー

クライアントプロファイル：フルクライアント

フルクライアント

- このクライアントタイプは、特定のリソース所有者に代わって、保護されたリソースにアクセスするために、ユーザーの権限の委任を必要とするクライアント。
- このクライアントは、リソース所有者が認可サーバーの認可エンドポイントと対話するのを容易にするために、別のWebアプリと対話することが可能である。
- このクライアントの考慮事項は下記の通り。

項番	考慮事項
1	このクライアントは、OAuth 2の認可コードフローを使用しなければならない。
2	ユーザは、認可エンドポイントに対して認証を行わなければならない。
3	これらのクライアントは、一意の公開鍵に関連付けられな ければならない。
4	アクセスリクエストのセキュリティパラメータがそれを許す場合、リフレッシュトークンをリクエストして発行してもよい

クライアントプロファイル：ネイティブクライアント

ネイティブクライアント

- このクライアントタイプは、特定のリソース所有者に代わって保護されたリソースにアクセスするために、ユーザーの権限の委任を必要とするクライアント。
- これらのクライアントはリソース所有者の認可サーバの認可エンドポイントとの相互作用を容易にするために、WEBアプリとの対話が可能である。
- 特に、このクライアントタイプはリソース所有者のデバイスでネイティブに実行される。また多くの場合、異なる環境で動作し異なるエンドユーザーに対して同時に実行されるソフトウェアのインスタンスが多数発生する。
- このタイプのクライアントの考慮事項は下記の通り。

項番	考慮事項
1～4	フルクライアントの考慮事項と同一
5	ネイティブクライアントは、ダイナミッククライアント登録を使用して、各インスタンスごとに別個のクライアント IDを取得しなければならない
6	トークンエンドポイントへの呼び出しを保護するため、クライアント鍵を使用しなければならない
7	動的登録を使用するネイティブアプリケーションは、デバイス上で一意の公開鍵と 秘密鍵のペアを生成し、その公開鍵の値を認可サーバーに登録すべきである
8	認可サーバーは、登録プロセスの一部としてクライアントに公開鍵と秘密鍵のペアを発行してもよい
9	そのような場合、認可サーバーはプライベート鍵のコピーを破棄しなければならない
10	ライアントの信用証明書は、クライアントソフトウェアのインスタンス間で共有してはならない
11	パブリックネイティブクライアントは、S256コードチャレンジメカニズムを使用して PKCEを使用しなければならない
12	機密性の高いネイティブクライアントも同様にPKCEを使用してもよい

クライアントプロファイル：フロントエンドクライアント

フロントエンドクライアント

- このクライアントタイプは、特定のリソース所有者に代わって保護されたリソースにアクセスするために、ユーザーの権限の委任を必要とするクライアント。
- これらのクライアントは Web ブラウザ内に埋め込まれており、システム間でアクティブなセッションを共有する。
- これらのクライアントは、OAuth 2.0のインプリシットフローを使用する。
- このフローは、バックエンドのサーバを持たないJavaScriptクライアントなど、フロントエンドのクライアントにのみ適している。
- このタイプのクライアントの考慮事項は下記の通り。

項番	考慮事項
1	ユーザは、認可エンドポイントに対して認証を行わなければならない。
2	リフレッシュトークンを要求したり、発行されたりしてはならない。
3	アクセストークンは短命でなければならない。
4	アクセストークンは、クライアントとのユーザーの認証されたセッションが期限切れになったときに破棄されるべきである

クライアントプロファイル：ダイレクトアクセスクライアント

ダイレクトアクセスクライアント

- このプロファイルは、保護されたリソースに直接接続し特定のリソース所有者に代わって行動しないクライアントに適用される。
- これらのクライアントは、クライアントクレデンシャルを使用してトークンエンドポイントにリクエストを送信し、応答でアクセストークンを取得することで、OAuth 2.0 のクライアントクレデンシャルフローを使用する。
- このプロファイルには認証済みユーザは含まれないため、このフローは開発者鍵を使用するような信頼されたアプリケーションにのみ適してる。
- このタイプのクライアントの考慮事項は下記の通り。

項番	考慮事項
1	リフレッシュトークンを要求したり、発行されたりしてはならない

フルクライアントとフロントエンドクライアントが、認可エンドポイントにリクエストする際における考慮事項

認可エンドポイントへのリクエスト

- 認可エンドポイントへのリクエストを行うフルクライアントとフロントエンドクライアントは下記の事項を考慮する必要がある。

項番	考慮事項
1	ステートパラメータに少なくとも128ビットのエントロピーを持つ、予測不可能な値を使用しなければならない。
2	クライアントは、リダイレクトURIに戻るときにステートパラメータの値を検証しなければならない。
3	state値がユーザーの現在のセッションに安全に関連付けられて いることを保証しなければならない。
4	クライアントは完全なリダイレクトURIを含まなければならない。
5	クライアントは認可リクエストに完全なリダイレクトURIを含まなければならない。
6	オープンリダイレクトや他のインジェクション攻撃を防ぐために、認可サーバーは、登録された値との直接文字列比較を使用して、リダイレクトURIを完全をマッチさせなければならない。
7	無効なリダイレクトURIまたは欠落したリダイレクトURIを持つリクエストを拒否しなければならない。

フルクライアントとフロントエンドクライアントが、トークンエンドポイントにリクエストする際における考慮事項

トークンエンドポイントへのリクエスト

- トークンエンドポイントへのリクエストについて、フルクライアントおよびダイレクトアクセスクライアントは、下記の事項を考慮する必要がある。

項番	考慮事項
1	JWT Profile for OAuth 2.0 Client Authentication and Authorization Grants および OpenID Connect Core で定義されているprivate_key_jwtメソッドで定義されている JWT Assertionを使用して、トークンエンドポイントを認証しなければならない。
2	アサーションは、以下のクレームを使用しなければならない。 iss：トークンを作成したクライアントのクライアントID sub：トークンを作成したクライアントのクライアントID aud：認可サーバのトークンエンドポイントのURL iat：クライアントがトークンを作成した時刻 exp：トークンが無効であるとみなされる期限時間 jtl：この認証のためにクライアントが生成した一意の識別子。
3	JWT Assertionは、クライアントの秘密鍵を使用してクライアントが署名しなければならない。
4	認可サーバーは、RS256署名方式をサポートしなければならない。
5	JSON Web Algorithms(JWA)仕様に記載されている他の非対称署名方式を使用してもよい。
6	ネイティブクライアント(PKCEを使用する)は、code_verifierをトークンエンドポイントに送らなければならない。
7	ネイティブクライアントは、このセクションで説明されているように公開鍵を使用して認証してもよい。
8	他の認証メカニズムを使用してはならない。

クライアント登録の際における考慮事項

クライアント登録

- クライアント登録について、下記の事項を考慮する必要がある。

項番	考慮事項
1	すべてのクライアントは認可サーバーに登録しなければならない。
2	認可サーバーに複数のインスタンスを登録するクライアントは、それぞれが一意のクライアント識別子を受け取らなければならない。
3	認可コードまたはインプリシットのグラントタイプを使用するクライアントは、完全なリダイレクトURIを登録しなければならない。
4	認可サーバーは、厳格な文字列比較を使用して認可エンドポイントでクライアントが与えたリダイレクトURIを検証しなければならない。
5	クライアントは、リダイレクトURIが以下のいずれかであることを確実にすることでリダイレクトURIに返される値を保護しなければならない。 １．TLS保護されているウェブサイト上にホストされている。 ２．クライアントのローカルドメインにホストされている。 ３．クライアント固有の非リモートプロトコルURIでホストされている。
6	クライアントは2つ以上のカテゴリのURIを持ってはならない。
7	異なるドメインに複数のリダイレクトURIを持つべきではない。
8	クライアントはリダイレクトURIに渡された値を他の任意のURIやユーザー提供のURIに転送してはならない。

クライアントキーに関する考慮事項

クライアントキー

- クライアントキーについて、下記の事項を考慮する必要がある。

項番	考慮事項
1	認可コードグラントを使用するフルクライアント、またはクライアントクレデンシャルグラントを使用するダイレクトアクセスクライアントは、トークンエンドポイントへの認証に使用する公開鍵と秘密鍵のペアを持たなければならない。
2	これらのクライアントは、公開鍵をjwksフィールドに直接送信するか認可サーバーが到達可能でなければならない。
3	jwks_uriを登録することで、クライアント登録メタデータに公開鍵を登録しなければならない。
4	可能であればクライアントはjwks_uriを使用することが推奨される。
5	クライアントのjwksフィールドまたはjwks_uriから利用可能なコンテンツは、JSON Web Key set (JWKセット)形式の公開鍵を含まなければならない。
6	認可サーバは、クライアントの登録されたjwks_uriドキュメントの内容を検証し、JWKセットが含まれていることを検証しなければならない。

保護されたリソースへリクエストする際の考慮事項

保護されたリソースへのリクエスト

- 保護されたリソースへリクエストする際には、下記の事項を考慮する必要がある。

項番	考慮事項
1	Authenticationヘッダーで渡されたベアラートークンを送るべきである。
2	クライアントは、RFC6750のform-parameterまたはquery-parameterメソッドを使用してもよい。
3	jwks_uriを登録することで、クライアント登録メタデータに公開鍵を登録しなければならない。
4	認可リクエストはTLS上で行われなければならない。
5	クライアントは保護されたリソースサーバーの証明書を検証しなければならない。

第3章では、RFC6749のSecurity ConsiderationsとOAuth Threat Model Documentを参照した上で、認可サーバが準拠すべき事項について記載されている。

認可サーバのプロファイル

- 全ての認可サーバは、RFC6749のSecurity Considerationsのセクションにある推奨事項と、OAuth Threat Model Documentにある推奨事項を準拠しなければならない。
- 認可サーバはOAuthエンドポイントの間の通信をTLSを使用して保護しなければならない。
- また下記の項目に関して、考慮事項が詳細に記載されている。
    1. クライアントとの接続
    2. 保護されたリソースとの接続
    3. トークンのライフタイム
    4. スコープ

1. 認可サーバとクライアントとの接続における、グラントタイプとクライアント認証の考慮事項

グラントタイプについての考慮事項

- 認可サーバはグラントタイプについて下記の事項について考慮する必要がある。

項番	考慮事項
1	認可サーバーは、authorization_code、 implicit、および client_credentials グラントタイプをサポートしなければならない。
2	認可サーバーは登録された各クライアントを、単一のグラントタイプに限定しなければならない。
3	複数の動作モードを持つクライアントは、モードごとに別々のクライアントIDを持たなければならない。

クライアント認証

- 認可サーバはクライアント認証について下記の事項について考慮する必要がある。

項番	考慮事項
1	認可サーバーは、認可コードとクライアントクレデンシャルのグラントタイプについて、クライアント認証を実施しなければならない。
2	認可サーバーは、認可コードおよびインプリシットグラントタイプのすべてのリダイレクトURIを検証しなければならない。

1. 認可サーバとクライアントとの接続における、クライアント登録の考慮事項

クライアント登録について

- 認可サーバはクライアント登録について下記の事項について考慮する必要がある。

項番	考慮事項
1	認可サーバーは動的クライアント登録をサポートしなければならない。
2	クライアントは、認可コードまたはインプリシットグラントタイプのために動的クライアント登録プロトコルを使用して登録してもよい。
3	クライアントは、クライアントクレデンシャルグラントタイプに動的に登録してはならない。
4	認可サーバーは、動的に登録されたクライアントが利用できるスコープを制限してもよい。
5	認可サーバーは、クライアントが認可画面で動的に登録されたことをエンドユーザーに通知しな ければならない。
6	認可サーバーは、[RFC7591]で述べられているように、信頼できる登録エンティティからクライアントソフトウェア開発者に発行された、署名されたソフトウェアステートメントを受け入れてもよい。
7	ソフトウェアステートメントは、実行される見込みで、動的に登録され、実行時に別個に認可される同じクライアントソフトウェアの、多数のインスタンスを結びつけるために使用できる。ソフトウェアステートメントには、以下のクライアントメタデータパラメータを含まなければならない。 redirect_uris grant_types jwks_uri or jwks client_name client_uri

1. 認可サーバとクライアントとの接続における、クライアント同意とDiscoveryの考慮事項

クライアント同意について

- 認可サーバはクライアント同意について下記の事項について考慮する必要がある。

項番	考慮事項
1	対話型の同意ページでエンドユーザーに促す際には、認証サーバーがエンドユーザーに下記の事項を示さなければならない。 １．クライアントが動的に登録されているか、信頼できる管理者によって静的に登録されているかどうか。 ２．クライアントがソフトウェア・ステートメントに関連付けられているかどうか、その場合はソフトウェア・ステートメントの信頼できる発行者に関する情報を提供すること。 ３．クライアントがどのようなアクセスを要求しているか（スコープ、ターゲットリソースなど）。

Discoveryについて

- 認可サーバはDiscoveryについて、下記の事項について考慮する必要がある。

項番	考慮事項
1	認可サーバは、OAuthプロトコルに関連するOpenID ConnectのDiscoveryエンドポイントを提供しなければならない。 issuer（JWT発行者の識別子） authorization_endpoint（認可エンドポイントのURL） token_endpoint（トークンエンドポイントのURL） introspection_endpoint（イントロスペクションエンドポイントのURL） revocation_endpoint（リボケーションエンドポイントのURL） jwks_uri（JWKフォーマットのサーバ公開鍵のURL）
2	クライアントと保護されたリソースはこの発見情報をキャッシュすべきである。
3	サーバは、HTTPヘッダを通じてキャッシュ情報を提供し、キャッシュを少なくとも1週間有効にすることが推奨される。
4	サーバは、JWK セット形式で公開鍵を提供しなければならない。
5	鍵は”kid”（トークンに署名するために使用されたキーペアのキーID）,”kty”（キータイプ）,”alg”（キーで使われているアルゴリズム）のフィールドを含めなければならない。

1. 認可サーバとクライアントとの接続における、Revocation、PKCE、リダイレクトURIの考慮事項

Revocationについて

- 認可サーバはRevocationについて下記の事項について考慮する必要がある

項番	考慮事項
1	トークンの失効を要求するクライアントがトークンが発行されたクライアントであり、クライアントがトークンを失効する許可を持っており、トークンが失効可能である場合、認証サーバーはトークンを失効させなければならない。
2	クライアントは、トークンを失効した後、直ちにトークンを破棄し、再び使用しないようにしなければならない。

PKCE(Proof Key for Code Exchange)について

- 認可サーバはPKCEについて下記の事項について考慮する必要がある

項番	考慮事項
1	認可サーバは、S256コードチャレンジ方式のサポートを含め、認可コードフローの PKCE(Proof Key for Code Exchange)拡張をサポートしなければならない。
2	認可サーバは、HEARTクライアントがプレーンコードチャレンジ方式を使用することを許可してはならない。

リダイレクトURIについて

- 認可サーバはリダイレクトURIについて下記の事項について考慮する必要がある

項番	考慮事項
1	認可サーバーは、認可リクエスト中に提示されたリダイレクトURIとクライアントの登録されたリダイレクトURIを正確に比較しなければならない。

2. 認可サーバと保護されたリソースとの接続における、JWTベアラートークンの考慮事項

JWTベアラートークンに関する考慮事項

- JWT に含まれる情報は、保護されたリソースが追加のネットワーク呼び出し無しでトークンの完全性を迅速にテストできるようにし、保護されたリソースがどの認可サーバでトークンを発行したのかを特定できるようにすることを目的としている。
- 保護されたリソースはJWTベアラートークンに関して下記の事項について考慮する必要がある。

項番	考慮事項
1	サーバーは、最低でも以下のクレームを持つJWTとしてトークンを発行しなければならない。 iss（トークンの発行者を示すURL） sub（Issuerごとにユニークで再利用されないエンドユーザの識別子） aud（トークンが有効な保護されたリソースの識別子を含む配列） exp（有効期限、この時間を過ぎた場合はトークンを無効にしなければならない） iat（JWTの発行時刻） jtl（少なくとも128ビットのエントロピーを持つ一意のJWTトークンID値） azp（このトークンを発行されたクライアントのクライアントID）
2	アクセストークンはJWSで署名されなければならない。
3	認可サーバーは、トークンに対してRS256署名方式をサポートしなければならない。
4	IANA JSON Web Signatures and Encryption Algorithmsレジストリで定義されている他の非対称署名方式を使用してもよい。
5	JWS ヘッダ”kid”フィールドを含まなければならない。
6	リフレッシュトークンは、同じ公開鍵を使用してJWSで署名され、アクセストークンと同じクレームのセットを含むべきである。
7	認可サーバーは、JWEを使用してアクセストークンとリフレッシュトークンを暗号化してもよい。
8	暗号化されたアクセストークンは、保護されたリソースの公開鍵を使用して暗号化されなければならない。
9	暗号化されたリフレッシュトークンは、認可サーバーの公開鍵を使用して暗号化されなければならない。

2. 認可サーバと保護されたリソースとの接続における、トークンインストロスペクション考慮事項

イントロスペクションに関する考慮事項

- トークンイントロスペクションは、保護されたリソースがトークンに関するメタデータを認証サーバに問い合わせることを可能にする。
- 認可サーバはトークンイントロスペクション仕様で定義されている以下のフィールドを含むトークンを表すJSONオブジェクトでイントロスペクション要求に応答する。

項目	説明
active	このトークンがこの認可サーバーで現在アクティブであるかどうかを示すブール値
scope	OAuth2.0のスコープ値をスペースで区切ったリスト
exp	このトークンの有効期限
sub	このトークンを承認したユーザーを一意に識別する識別子
client_id	ことトークンを要求したクライアントのクライアントID

- 認可サーバはトークンインストスペクションに関して下記の事項について考慮する必要がある。

項番	考慮事項
1	認可サーバーは、revocationエンドポイントとイントロスペクションエンドポイントの両方に認証を要求しなければならない。
2	インストロスペクションエンドポイントを呼び出す保護されたリソースは、サーバーに登録された他のOAuthクライアントとは異なる別個のクレデンシャルを使用しなければならない。
3	保護されたリソースはイントロスペクションエンドポイントからの応答をキャッシュしてもよい。

3. 認可サーバのトークンのライフタイムについての考慮事項

トークンのライフタイムに関する考慮事項

- HEARTプロファイルでは、異なるタイプのクライアントに発行される、異なるタイプのトークンの推奨寿命に関していくつかの考慮事項を記載している。

項番	考慮事項
1	特定のアプリケーションは、異なるライフタイムでトークンを発行してもよい。
2	アクティブなトークンはいつでも取り消せる。
3	認可コードグラントタイプを使用するクライアントの場合、アクセストークンの有効期限は1時間以内にするべき。
4	リフレッシュトークンの有効期限は24時間以内であるべきである。
5	インプリシットグラントタイプを使用するクライアントの場合、アクセストークンの有効期限は15分を超えてはならない。
6	クライアントクレデンシャルグラントタイプを使用するクライアントの場合、アクセストークンの有効期限は 6 時間を超えないようにすべきである。

4. 認可サーバのスコープについての考慮事項

スコープに関する考慮事項

- スコープは、クライアントからの要求、リソースオーナからの提供、保護されたリソースからの強制を可能にする個々の権限を定義する。
- 認可サーバ・保護されたリソースは、スコープに関して下記の事項を考慮する必要がある。

項番	考慮事項
1	認可サーバは、認可要求で要求されたスコープのセットを指定しない場合に使用されるデフォルトのスコープ値を定義し、文書化すべきである。
2	様々な保護されたリソース全体での一般的な使用を容易にするために、認証サーバは、クライアントや保護されたリソースが登録時に任意のスコープ文字列を使用することを許可するなど、実行時に任意のスコープ値の使用を許可すべきである。
3	認可サーバは、動的に登録されたシステムによる特定のスコープの使用を制限してもよい。

保護されたリソースがクライアントと認可サーバに接続する際の考慮事項

クライアントとの接続

- 保護されたリソースがクライアントと接続する際の考慮事項は下記の通り。

項番	考慮事項
1	保護されたリソースは、[RFC6750]で述べられているように、authorizationヘッダーで渡されたベアラートークンを受け入れなければならない。
2	保護されたリソースは、formパラメータまたはqueryパラメータメソッドで渡されたベアラートークンも受け入れてもよい。
3	保護されたリソースは、そのリソースへのアクセスに必要なスコープを定義し、文書化しなければならない。

認可サーバとの接続

- 保護されたリソースが認可サーバと接続する際の考慮事項は下記の通り。

項番	考慮事項
1	保護されたリソースは、JWT、トークンイントロスペクション、またはそれらの組み合わせを使用してアクセストークンを解釈しなければならない
2	保護されたリソースは、トークンの中に aud(オーディエンス)クレームが存在する場合、それが保護されたリソースの識別子を含むことを確認しなければならない。
3	保護されたリソースは、トークンに関連付けられた権利がリソースへのアクセスを許可するのに十分なものであることを確認しなければならない。
4	保護されたリソースは、有効なトークンを受け入れる認可サーバーを制限しなければならない。
5	リソースサーバーは、信頼できるサーバーのホワイトリスト、ダイナミックポリシーエンジン、またはその他の手段を使用してこれを達成してもよい。

OAuth所有証明トークンを使うことで、ここまでHEARTプロファイルよりもさらに高いセキュリティレベルを実現することが可能となる。

Advanced OAuth Security Options

- HEARTプロファイルでは、開発者、システム管理者、エンドユーザにとっての実装の容易さと使いやすさを維持しつつ、幅広いユースケースに適したセキュリティレベルを提供している。
- この章では、実装の手間と使いやすさを犠牲にしても、リスクが高いために追加の制御を使用することが求められる場合に採用できる、追加のセキュリティ対策がいくつか紹介されている。

トークンの所有証明

- OAuth 所有証明トークンは、現在 Internet Engineering Task Force (IETF) OAuth作業部会で一連の草案で定義されている。
- ベアラートークンはトークンを所持している人なら誰でも使用できるが、所有証明トークンはクライアントに発行された、あるいはすでに所持している特定の対称鍵または非対称鍵に関連付けがされており、トークンと鍵の関連付けは保護されたリソースにも伝わる。
- クライアントが保護されたリソースにトークンを提示する際には、対応する鍵を所有していることを証明する必要がある。(例えば、リクエストの暗号化ハッシュや署名を作成するなど)
- 所有証明トークンを使うことで、不正な攻撃者によるアクセストークンの傍受を伴う多くの攻撃を防ぐことが可能となる。

HEARTのプロファイルを利用するにあたってセキュリティ面で考慮すべき事項がSecurity Considerationsとしてまとめられている。

Security Consideration

- すべてのトランザクションは、TLSで保護されなければならない。
- すべてのクライアントは、RFC6749のSecurity Considerationsのセクションにある適用可能な推奨事項と、OAuth 2.0 Threat Model and Security Considerationsのドキュメントにある推奨事項に準拠しなければならない。

項番	考慮事項
1	保護されたリソースは、[RFC6750]で述べられているように、authorizationヘッダーで渡されたベアラートークンを受け入れなければならない
2	保護されたリソースは、formパラメータまたはqueryパラメータメソッドで渡されたベアラートークンも受け入れてもよい
3	保護されたリソースは、そのリソースへのアクセスに必要なスコープを定義し、文書化しなければならない

4. Health Relationship Trust Profile for Fast Healthcare Interoperability Resources (FHIR) OAuth 2.0 Scopes

FHIRは、HL7(医療情報交換のための標準規格)の医療データ交換と医療情報モデル化の標準規格に基づき作成された新しい仕様である。

FHIR（Fast Healthcare Interoperability Resources）概要

- 現代医療の背景として電子カルテの普及や医療の自動化の促進のために、電子カルテが利用可能であることや、医療データの構造化が求められている。
- FHIRは、医療アプリケーション間のデータ交換のための厳密なメカニズムや、大多数の共通のユースケースを満たすリソースのベースセットなどの仕様の策定を行っている。
- FHIRリソースは、ほとんどのパターンで共有されている主要な情報セットのコンテンツと構造を定義することを目的としている。

HEART for FHIR OAuth 2.0 Scopesでは、FHIRで使用されるスコープをプロファイル化し、ヘルスケアドメインに適用可能な方法で構造化する。

HEART Scope 概要

- OAuth 2.0のベースラインのセキュリティを向上させより高い相互運用性を提供するために、FHIRで使用されるOAuth 2.0のスコープをプロファイル化し、ヘルスケアドメインに適用可能な方法で構造化したものである。

HEART Scope 詳細

- 本仕様では、スコープ値は、パーミッションのタイプ、リソースのタイプ、およびアクセスのタイプを記述した下記のような複合文字列である。
  - **scope := permission/resource.access**
- パーミッションタイプは要求されるアクセスが単一の患者記録に対するものか、一括セットの患者記録に対するものかを示す。
- リソースタイプはFHIRリソースで利用可能な情報の種類を示す。
- アクセスタイプは特定のリソースに対して、読み込み・書き込み等のアクションが可能かを定める。
- また、リソースの所有者が意思表示できない場合にアクセスが要求されることを示す特殊なスコープとして「btg」というスコープが定義されている。

本仕様では、FHIR versionSTU3から以下のリソースがリストアップされている。

リストアップされたFIHRリソース一覧

項目	説明
Patient	患者に関する人口統計学的情報
MedicationRequest	患者への投薬依頼についての情報
MedicationDispense	患者への薬の供給に関する情報
MedicationAdministration	薬の服用等に関する情報
ObservationMedicationStatement	医療従事者が行う観察についての情報
Appointment	予約についての情報
AllergyIntolerance	アレルギーと薬物不耐性に関する情報
Condition	患者の状態に関する情報
Immunization	患者の予防接種に関する情報
CarePlan	患者のケアプランに関する情報
*	与えられたコンテキスの下で、利用可能なすべてのリソースを表すワイルドカード

NRI SECURE