

第 3 章

FIDO と OpenID Connect の関係性について

こんにちは@super_reader です。普段は認証や ID 連携のことをやっており、その中でも主に FIDO(特に WebAuthn) を扱っている Web エンジニアです。
今回 FIDO に関してなにかネタはないかと考えたときに、自分に取り組んでいる認証と ID 連携の仕組みについて関連付けたものを書いたら、読んでくださる方の役にたつのではと思いました。

そこで、今回は「FIDO と OpenID Connect の関係性」について書きたいと思います。私自身、ID 連携に関して知識をかじった後に FIDO などの認証の世界を学んでいった経緯もあり、それぞれの仕様について触れてきた立場から、簡単にはなりますが、それぞれの仕様がどのように関連しているのかを解説したいと思います。

今回話したい題材は以下になります。

1. OpenID Connect にとっての FIDO とは？
2. OpenID Connect と FIDO での Relying Party について
3. FIDO と Self-issued について

それではさっそく FIDO と OpenID Connect の関係を見てみましょう。

3.1 OpenID Connect にとっての FIDO とは？

まずは OpenID Connect(OIDC) に関して簡単に説明したいと思います。
OIDC は ID 連携の仕様の一つです。ID 連携とは連携先のサービス (Relying Party) に対して、ID やパスワードを渡すことなく、ユーザーの認証情報を提供して、OpenId Provider に保存されているユーザーデータ (属性情報) にアクセスすることが可能になる

第3章 FIDO と OpenID Connect の関係性について

仕組みのことを指します。

ID 連携の仕様の一つである OIDC は現在さまざまなサービスで使われている仕様です。また、OpenId Provider は認証と認可の機能、そして、サービスがほしい属性情報を持っているサービスになります。実際の企業でいいますと Google や Yahoo! JAPAN、LINE などの ID Provider(IdP) を指します。

簡単に ID 連携に関して説明をさせていただきましたが、OIDC に関して、それぞれの細かな仕様のここでの説明は割愛させていただきます。(全部書いてしまうと、とても長くなってしまいますので)

その代わり以下に OIDC の理解に役立ちそうなサイトのリンクを記載させていただきます。

- 一番分かりやすい OpenID Connect の説明
- OAuth2.0 & OpenID Connect 基礎
- OpenID Connect 入門

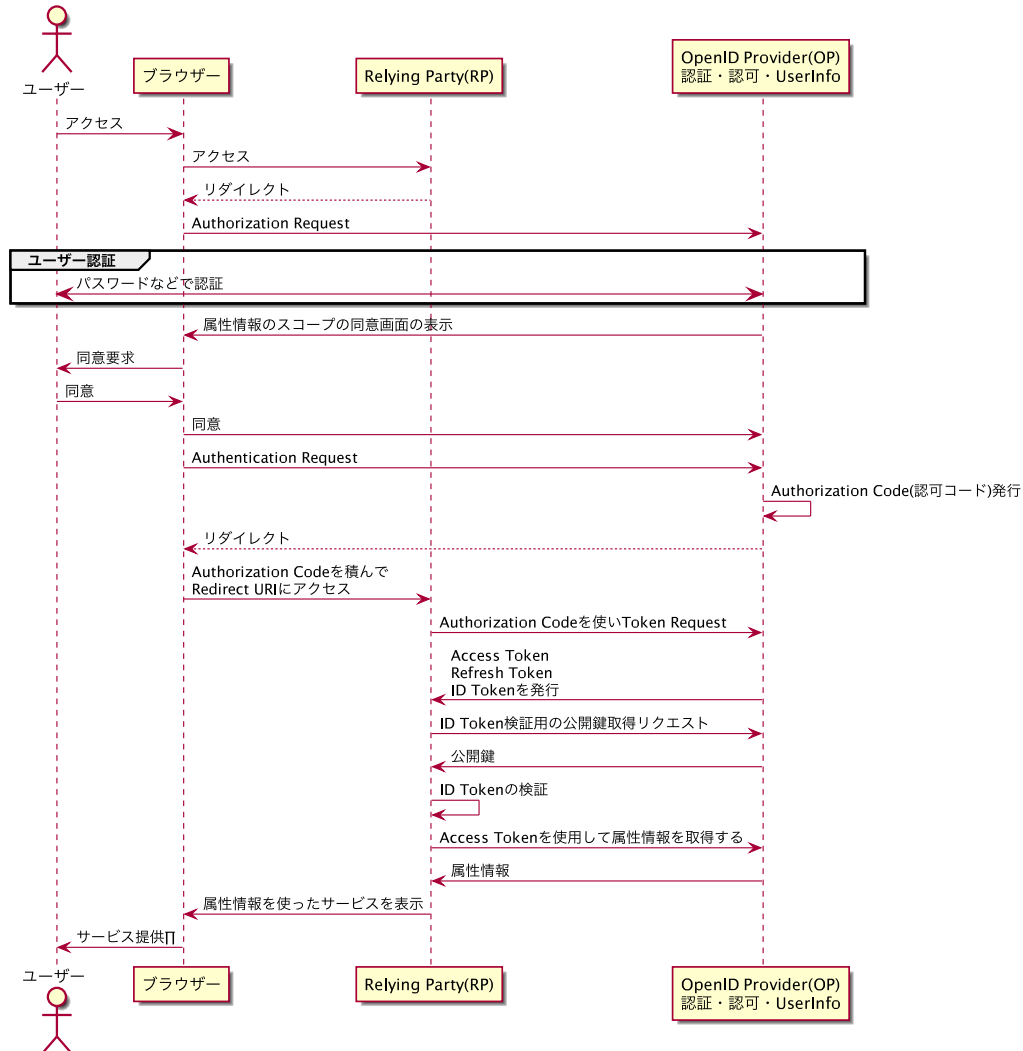
3.1.1 OpenID Connect と FIDO のフローを確認してみよう

さて、本題に入りたいと思います。

OIDC と FIDO の関係を考えていく上でまずは、OIDC がどのような流れで処理をされているのかを見てみましょう。

以下は OIDC のフローの一つである Authorization Code フローを簡単に書いたものになります。(図 4.1)

3.1 OpenID Connect についての FIDO とは？



▲図 3.1 Authorization Code フロー

図 4.1 は ID 連携を行いたいサービス (Relying Party) が、OpenId Provider(OP) の発行する認可用の Access Token を使い、属性情報を取得するためのフローになります。このフローの中でも書かれていますが Relying Party に認可コードを渡すために「ユーザー認証」を行っています。

OIDC の使用説明の中に OPTION ですが認証方法 (Authentication Methods References) について以下のような記述があります。

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。例として、パス

第 3 章 FIDO と OpenID Connect の関係性について

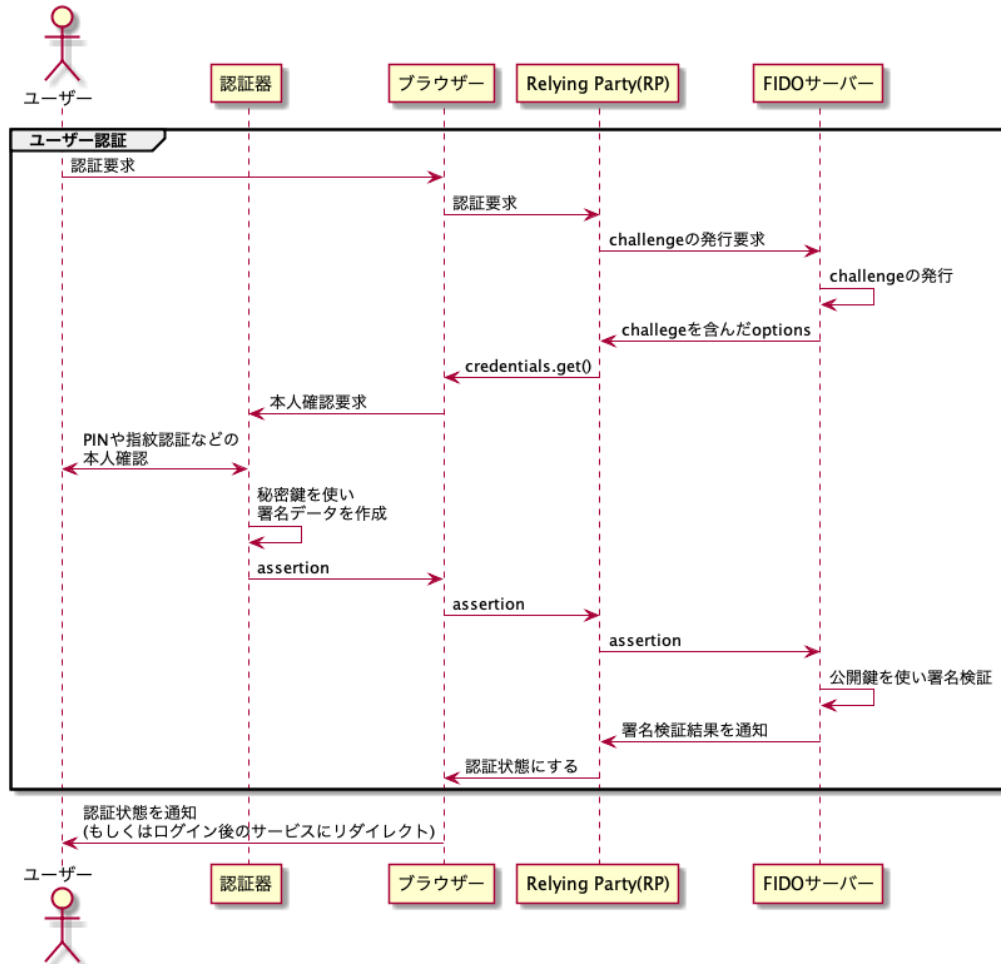
ワードと OTP 認証が両方行われたことを示すといったケースが考えられる。**.amr Claim** にどのような値を用いるかは本仕様の定めるところではない。この値の意味するところはコンテキストによって異なる可能性があるため、この Claim を利用する場合は、関係者間で値の意味するところについて合意しておくこと。**.amr** は大文字小文字を区別する文字列である。

このように、OIDC の仕様の中でも認証方法に関しては認証方法については仕様で細かく言及されていません。

これは言い換えると、認証方法に関しては OP の実装方法に委ねられているということが言えます。

さて、ここで FIDO(WebAuthn) の認証フローを見てみましょう。

3.1 OpenID Connect にとっての FIDO とは？



▲図 3.2 WebAuthn の認証フロー

わざとらしく書いてますが、図 4.2 を見てみると FIDO の仕様は認証だけで閉じています。

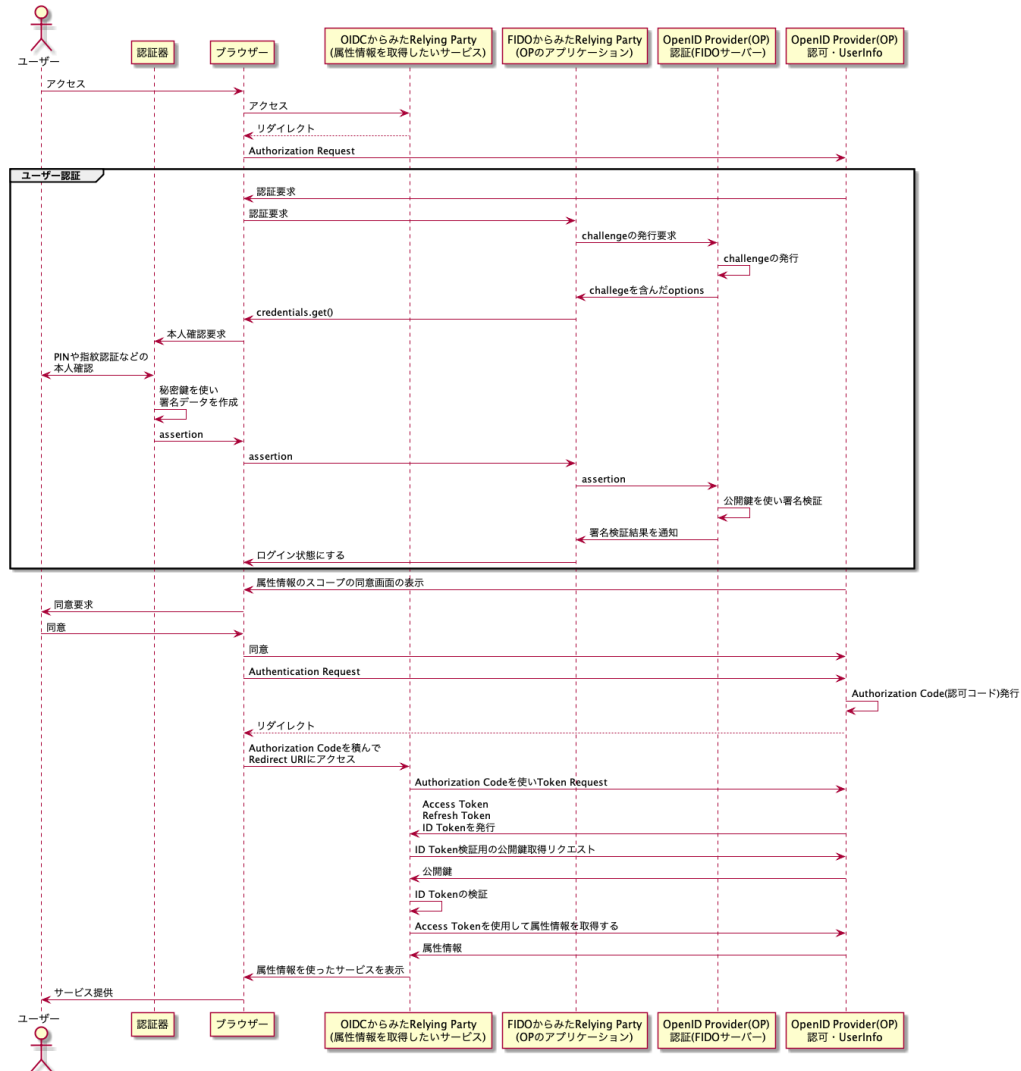
ここで気にしてほしいこととして、FIDO は認証の仕様ということです。

なので、OIDC の Authorization Code フローでも書かれていた「ユーザー認証」の部分に FIDO をそっくりそのまま入れ込むことができます。

これは、OP が FIDO 対応の認証手段を持っていた場合に OIDC のフローの中に FIDO を組み込むことが可能ということを示しています。

OIDC の Authorization Code フローに FIDO(WebAuthn) のフローを追加したものが図 4.3 です。

第3章 FIDO と OpenID Connect の関係性について



▲図 3.3 Authorization Code フローに FIDO のフローを追加した図

このように、ID 連携の仕様である OIDC と認証の仕組みである FIDO は同居することが可能であり、組み合わせることで ID 連携をしたいサービスでも FIDO を使った認証体験を実現することが可能となります。

3.1.2 実際の具体例を見てみよう

この仕組みが実際に行われているのが Yahoo!ID 連携です。Yahoo!ID 連携の画面フローをもとにどのような処理が行われているのかをたどっていきましょう。

今回は具体例としてヤマト運輸のクロネコメンバーズの会員登録を進めるまでの過程を見てみましょう。

図 4.4 はクロネコメンバーズのログイン画面です。Yahoo! JAPAN のログインボタンを押すことで ID 連携がスタートします。



▲図 3.4 RP の ID 連携ボタンがある画面

まずは OP 側の認証を行う必要があります。図 4.5 は OP である Yahoo! JAPAN のログイン画面です。未ログイン状態からログインするために ID を入力して、「次へ」ボタン

第3章 FIDO と OpenID Connect の関係性について

を押します。ここで使用するアカウントはすでに WebAuthn での登録フローが完了しているアカウントになります。



▲図 3.5 OP のログイン画面

「次へ」ボタンが押された瞬間に裏側では challenge の要求が走り、サーバーからは challenge を含んだ必要な options が返却されます。そして、credentials.get() が叩かれ、ブラウザに対して認証命令を送ります。この認証命令がブラウザに送られ、認証器（今回の場合は Android 内の認証器）が呼び出され、図 4.6 のローカル認証として指紋が要求されています。

3.1 OpenID Connect にとっての FIDO とは？



▲図 3.6 FIDO の認証で指紋が要求される画面

指紋認証に成功しましたら、サーバー側に対して assertion が送られ、署名されたデータを OP の FIDO サーバーに保存してある公開鍵で検証を行います。

ここで認証のフローは終わり、Token 発行のフローに入ります。

その後、認可コードを発行するために OP にアクセスします。このときに RP(ヤマト運輸) に対して、どのような属性情報を渡すのか同意を取る画面が現れ、ユーザーはどこまでの情報を RP に渡していいのかを確認します。

第 3 章 FIDO と OpenID Connect の関係性について

Y! XXXXXXXXXXさん

クロネコメンバーズサービス
<http://www.kuronekoyamato.co.jp/>

このサービスへの情報提供等 ([注意事項](#))

- お知らせ連携機能 ▲
このサービスの情報をヤフー株式会社提供のアプリの通知やWebサービス上でお知らせする機能をお客様に提供します。
- ユーザ識別子 ▼
- 姓・名・生年・性別 ▼
- メールアドレス ▼
- 住所情報 ▼
- 電話番号 ▼

[同意してはじめる](#)

[同意しない](#)

[プライバシー・規約・ヘルプ](#)
(C)Yahoo Japan

▲図 3.7 RP に対してどのような属性情報を渡すのかの同意画面

属性情報の同意が取れましたら、裏では図 4.3 のように各種 Token が払い出され、Access Token を使用して属性情報を取得します。

取得された属性情報はクロネコメンバーズの登録画面にプリセットされます。モザイクばかりになってしまいましたが、図 4.8 のように登録画面に OP(Yahoo! JAPAN) に保存されていた属性情報が埋め込まれた登録画面が現れます。

ヤマト運輸

新規登録

登録に必要な項目を入力してください。

お客様情報

メールアドレス 必須 ✓

XXXXXXXXXX@yahoo.co.jp

☒ スマホ・PC ☐ 携帯

※iPhoneやAndroid端末をご利用の場合は、必ず「スマホ・PC」を選択してください。

お名前（全角） 必須

姓 XXXXX 名 XXXXX

フリガナ（全角カナ）

セイ XXXXX メイ XXXXX

お電話番号（半角数字） 必須

XXXXXXXXXX

郵便番号

〒100-0001 東京都千代田区千代田

▲図 3.8 RP の登録時に属性情報がプリセットされている画面

このように ID 連携をすることで事前に必要な情報を取得することができ、ユーザーが登録をしやすような土台を RP(ヤマト運輸) はユーザーに提供することができます。

以上の具体例からも OIDC と FIDO は共存することが可能であることがわかります。

3.2 OpenID Connect と FIDO での Relying Party について

私が FIDO を一番最初に学んだときに一番最初に引っかかった部分がここでした。突然ですが、先程見てみた Authorization フローに FIDO のフローを追加した図 (図 4.3) をもう一度見てみましょう。

図 4.3 の中には Relying Party(RP) という単語が 2 回出てきます。OIDC と FIDO の両方で登場するこの単語なのですが、2つの仕様で指し示すサービスが違います。

OIDC の中では ID 連携を行いたいサービスを RP と呼び、FIDO では主に ID Provider(OIDC では OP となっている部分) のことを RP と呼びます。

同じ ID 関連の仕様なのに同じ単語で違う意味合いになっているのはなぜでしょうか？

この RP が二箇所に出てきてしまった問題を考えるにあたって、RP の元々の意味を考える必要が出てきます。

ID 関連の仕様で使われている RP の元々の意味を「認証 (本人検証) を委任しているサービス」と捉えることにより、この疑問は解消されるはずです。(ここではわかりやすく考えるため OpenID Provider = ID Provider と考えてください)

OIDC の場合

ID 連携を行いたいサービス (RP) が OP に対して認証 (本人検証) を委任している

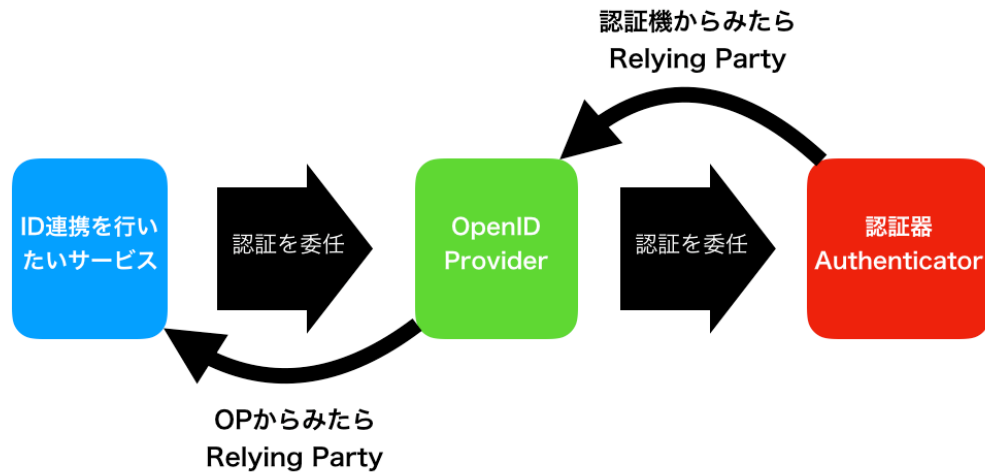
FIDO の場合

ID Provider(RP) が認証器 (Authenticator) に対して認証 (本人検証) を委任している

という風に解釈できます。

よくあるパスワードでのログインをする場合などにおいて ID Provider が認証 (本人確認) を委任することは基本的にはないのですが、FIDO の文脈では ID Provider が認証 (本人確認) を認証器 (Authenticator) に委任しています。

そのため OIDC の枠組みに FIDO を導入しようと思った場合、認証 (本人確認) は図 4.9 のように委任を繰り返します。



▲図 3.9 それぞれの RP の関係

FIDO を学び始めたとき、最初は同じ意味を指している単語だと思い、自分の解釈が間違っていると思い RP という単語で混乱していました。
しかし、RP という言葉の意味をよく考えてみると矛盾なく、同じ単語でサービスが違っているのかも理解することができます。

3.3 FIDO と Self-issued について

突然ですが、OIDC の仕様の中にも認証をどうするかについて書かれた **Self-issued** という仕様があります。

参考:https://openid.net/specs/openid-connect-core-1_0.html#SelfIssued
この仕様を簡単に説明すると

1. OIDC における認証部分において、OpenID Provider の認証機能をローカルデバイスの中に持っていき、認証を行う
2. 認証方法は公開鍵暗号方式を使ったもので、秘密鍵などのクレデンシャル情報はローカルデバイスの中に保存する

などが挙げられます。

実際に OIDC の枠組みを使い、Self-issued を取り入れている企業としてはリクルートが有名です。