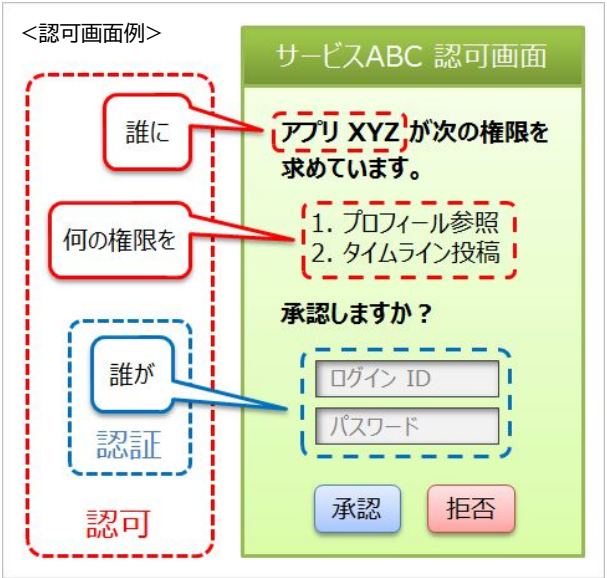
認証/認可説明資料

目次

認証/認可の違い

認証は「利用者が確かに本人であることを確認すること」
認可は「認証された利用者に権限を与えること」

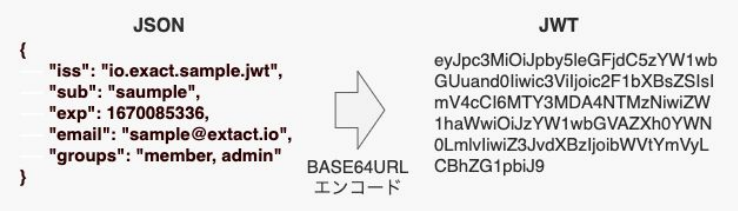


- 認証(Authentication)
  - 誰であるのか(Who he is)を扱う
  - **ユーザーの一意識別子を特定する処理**
- 認可(Authorization)
  - 誰が誰に何の権限を与えるか(Who grants what permissions to whom)を扱う
  - **認可処理にはその一部として認証処理が含まれている**

出典:
https://qiita.com/TakahikoKawasaki/items/f2a0d25a4f05790b3baa#

JWT(JSON Web Token)とは

HTTPヘッダやクエリパラメータに載せるためにJSON形式のデータを「URLセーフに」「コンパクトな形式」にしたもの



出典:
https://developer.mamezou-tech.com/blogs/2022/12/08/jwt-auth/

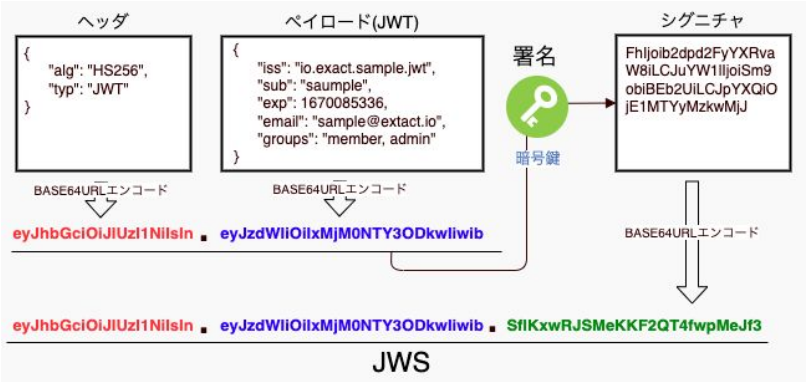
- URLセーフする方法
  - JSONデータをBASE64URLエンコードする
- コンパクトにする方法
  - よく使われるデータ項目の名称を省略形にすることでJSONのキー名を短くする

JWT自体は何ら認証とは関係なく、認証データをやり取りする伝達手段としてJWTが都合が良いために使われているだけ

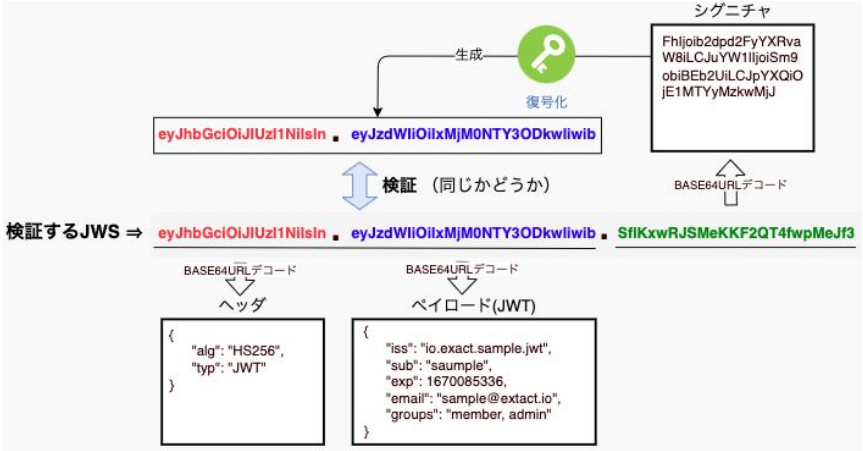
JWTを用いた改ざん防止技術

JWS(JSON Web Signature)を用いて改ざん防止の検証を行うことができる

<JWSの生成(JWSコンパクトシリアライゼーション)>



<JWSの検証>



出典:
https://developer.mamezou-tech.com/blogs/2022/12/08/jwt-auth/

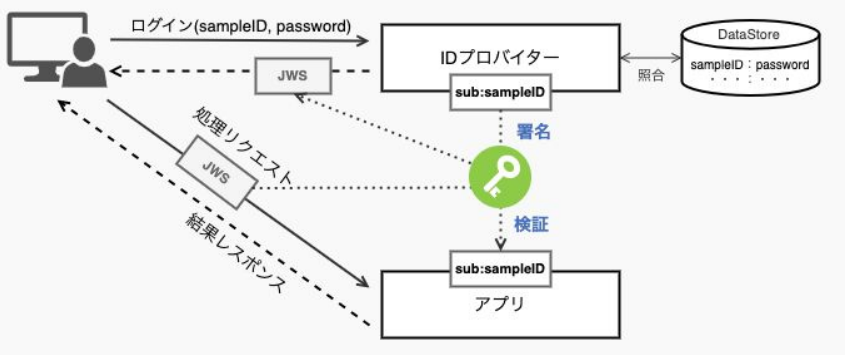
JWT関連のその他の技術

- JWE(JSON Web Encryption)　(RFC7516)
  - 暗号化を行う
- JWK(JSON Web Key) (RFC7517)
  - 暗号鍵を表現するJSONオブジェクト
- JWA(JSON Web Algorithms) (RFC7518)
  - JWE、JWS、JWKで使用する暗号化アルゴリズムの識別子を登録している

JWTによる認証

認証情報をのせたJWTをもとにJWS/JWE等の技術を用いてユーザー認証を行うこと

<JWT認証>



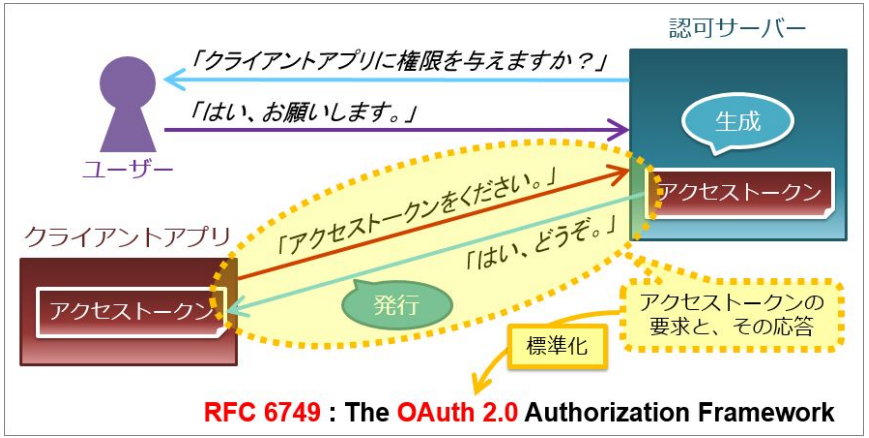
出典:
https://developer.mamezou-tech.com/blogs/2022/12/08/jwt-auth/

- JWTを使用するメリット
  - 認証処理をプレゼンテーションで完結することができ、DBアクセスが不要となることで**認証処理の性能が向上**
  - クライアントからリクエストの都度、認証されたJWTを送信してもらい、それを検証することでユーザをセキュアに識別できるため、**セッションでユーザ情報を維持する必要がない**

OAuthとは

アクセストークンの要求とその応答を標準化したもので認可の仕組みを指す

<OAuth動作図解>



出典:
https://qiita.com/TakahikoKawasaki/items/f2a0d25a4f05790b3baa#

1. クライアントアプリがアクセストークンを要求する
2. 認可サーバーはユーザーに確認を取る
3. ユーザーは許可をする
4. クライアントアプリにアクセストークンが発行される

OAuthの認可フロー

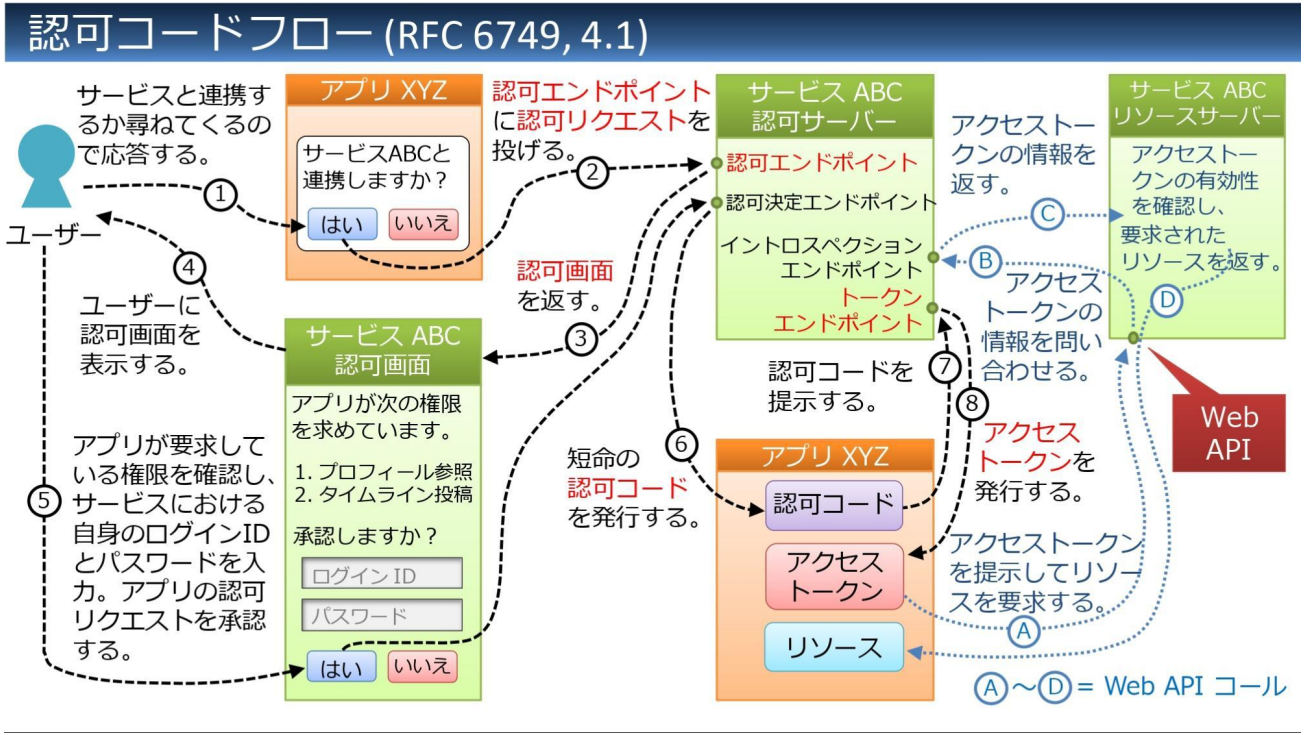
参考:
https://logmi.jp/tech/articles/322822

OAuth2.0では「認可エンドポイント」「トークンエンドポイント」二つの動作を定義しており、4つの認可フローが存在する

	認可フロー名	認可エンドポイント	トークンエンドポイント	特徴
1	認可コード	○	○	一時的に発行される認可コードを用いてアクセストークンを取得する
2	インプリシットフロー	○	×	認可エンドポイントから直接アクセストークンが発行される
3	リソースオーナー・パスワードクレデンシャルズ	×	○	ユーザーのIDとパスワードをクライアントアプリに直接渡す
4	クライアント・クレデンシャルズ	×	○	ユーザー認証なし．クライアントアプリ認証のみ必要．

認可コードフロー

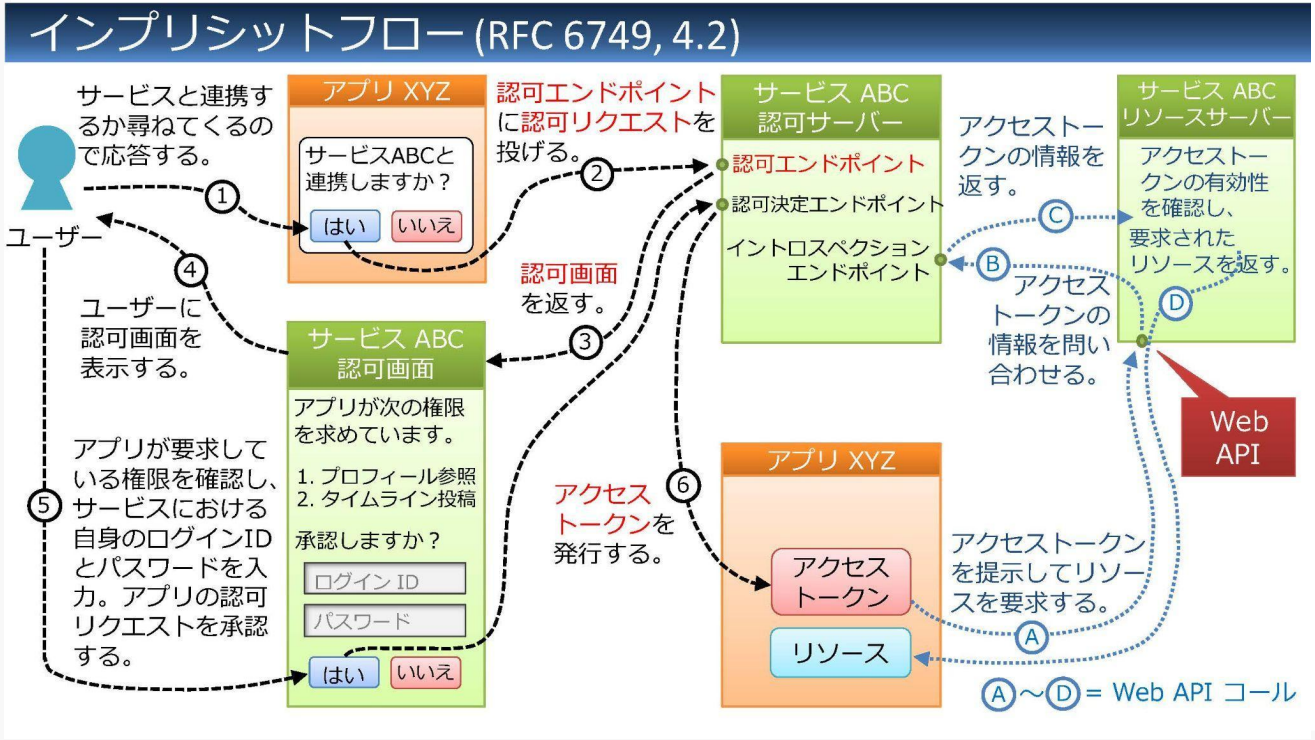
参考:
https://logmi.jp/tech/articles/322822



- OAuth2.0で定義されている点
  - ②
  - ⑥
  - ⑦
  - ⑧

インプリシットフロー

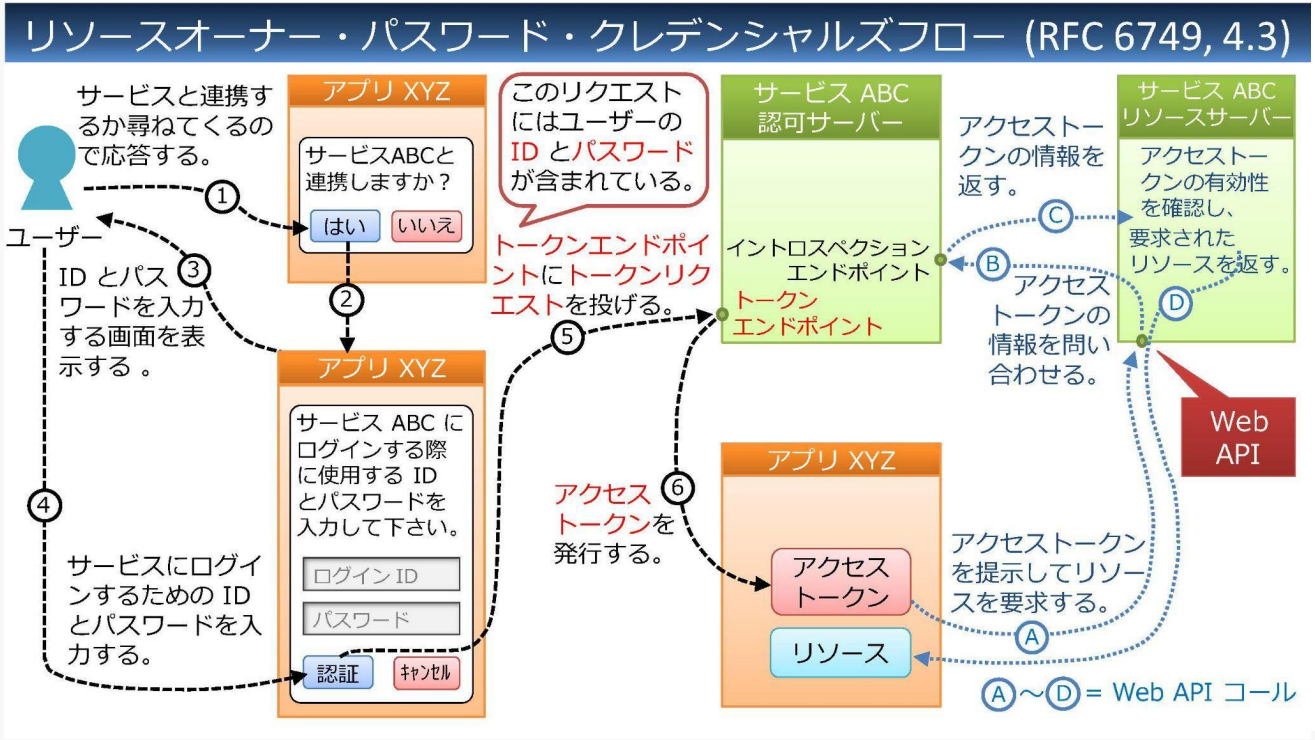
参考:
https://logmi.jp/tech/articles/322822



- OAuth2.0で定義されている点
  - ②
  - ⑥
- このフローは**もう使用しないように**推奨されている点に注意

リソースオーナーパスワードクレデンシャルズフロー

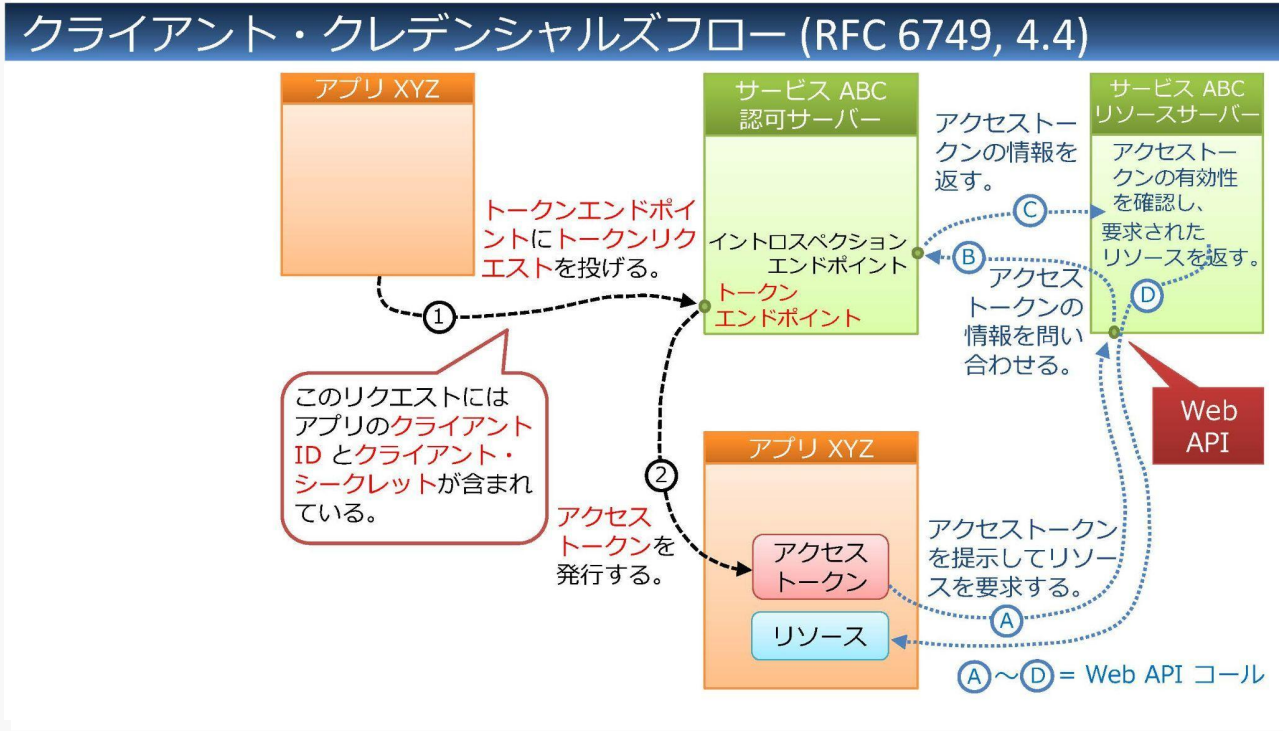
参考:
https://logmi.jp/tech/articles/322822



- OAuth2.0で定義されている点
  - ⑤
  - ⑥
- このフローは**もう使用しないように**推奨されている点に注意
  - OAuth2.0策定当時も移行用のためにこのフローが存在した

クライアントクレデンシャルズフロー

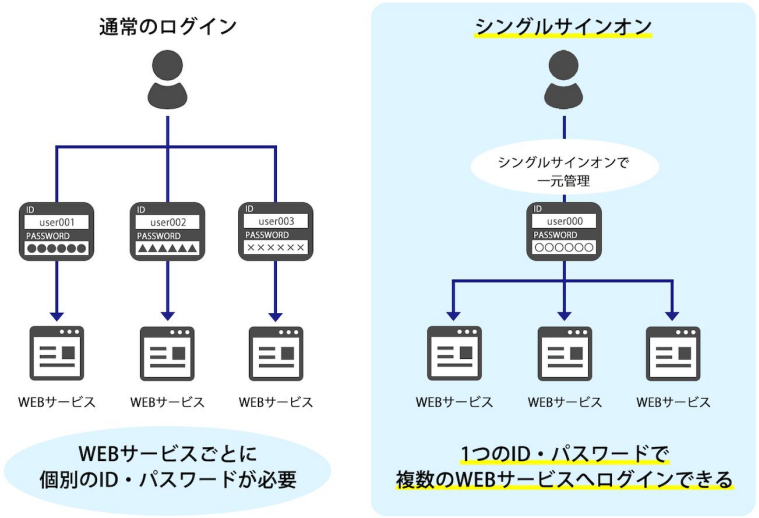
参考:
https://logmi.jp/tech/articles/322822



- OAuth2.0で定義されている点
  - ⑤
  - ⑥
- ユーザーが絡まない認証に使用される

SSO(Single Sign-On)とは

1度のユーザー認証によって複数のシステムの利用が可能になる仕組みを指す



出典:
https://www.splashtop.co.jp/knowhow/25/

- メリット
  - 管理の手間が省ける
  - 利便性の向上
- デメリット
  - 流出によるリスクの増大
- 代表的な種類
  - リバースプロキシ
  - ケルベロス認証
  - SAML(Security Assertion Markup Language)

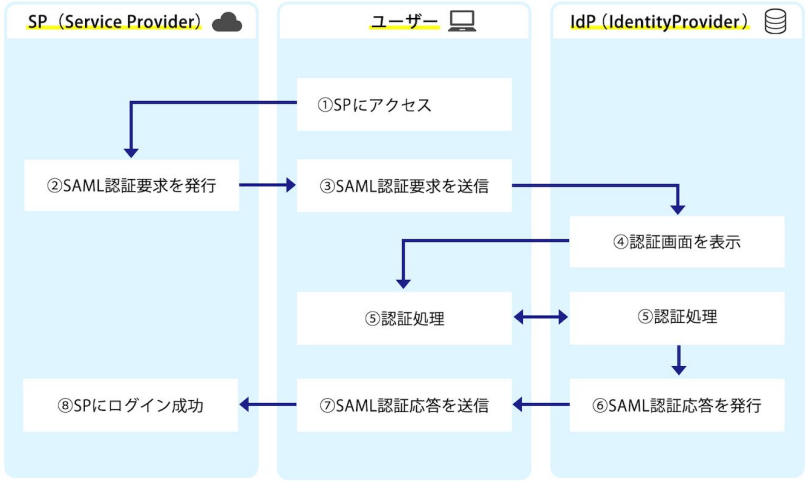
SAML(Security Assertion Markup Language)とは

参考:
https://boxil.jp/mag/a2950/#2950-3-1

異なるインターネットドメイン間でユーザー認証情報をやりとりするためのXMLベースの規格

- 用語
  - IdP(Identity Provider)
    - 認証を受け持つサーバー
  - SP(Service Provider)
    - 実際に利用するアプリ・インターフェース
  - Assertion
    - SAMLで使用する情報が含まれるトークン

<SAML認証フローの例(SP Initialized)>



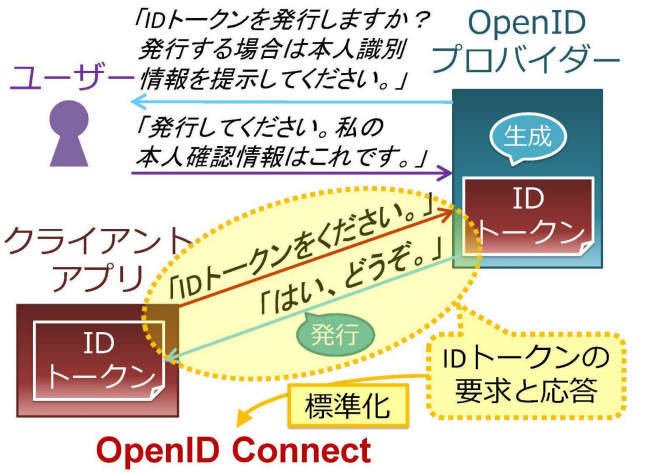
出典:
https://www.splashtop.co.jp/knowhow/26/

OIDC(OpenID Connect)とは

OAuth2.0の使用を拡張したものでアクセストークンに加え、IDトークンの発行もできるようにしたもの

- IDトークンとは
  - 認証情報に発行者の署名を加えたもの
- メリット
  - OAuthでは定義されていなかった認証部分について補強されたため, **ユーザー認証についても使用することができる**

<OIDC動作図解>



出典:
https://qiita.com/TakahikoKawasaki/items/498ca08bbfcc341691fe

SMS認証とは

参考:
https://www.onelogin.com/jp-ja/learn/what-is-mfa

MFAの一つで、ユーザ認証の際にID、パスワードに加えショートメッセージによるコードの入力等を求める認証方法

- MFA(多要素認証)とは
  - ユーザ名とパスワードだけでなく、**1つ以上の追加の検証要素を要求**しセキュリティ強化を図る認証方法のこと

- SMS認証のメリット
  - 比較的低コストに導入可能
  - セキュリティ強化を図れる

<SMS認証例(ワンタイムパスワード)>

出典:
https://www.smsnavi.com/kiso/certification/