学術機関におけるID連携とOpenID Connect

2024/09/17

富士榮　尚寛
OpenIDファウンデーションジャパン代表理事
OpenID Foundation eKYC and Identity Assurance WG共同議長
伊藤忠テクノソリューションズ株式会社 みらい研究所長

Challenging Tomorrow's Changes

本日のお話し（アジェンダ）

- はじめに
- 学術機関で利用するIdP/SPの傾向
- SAMLを知っている人向けのOpenID Connect
- OpenID Connectの基本
- OpenID Connect/OAuthの応用
- 機関IdPをOpenID Connect対応にするためには
- その他の話題

はじめに

- SAMLとOpenID Connectの比較や優劣について話すセッションではありません
- あくまで学術機関でOpenID Connectに対応した機関IdPを使うために必要な技術情報の提供が目的です

OpenID Connectの普及状況 by Perplexity

主要プラットフォームでの採用

- OpenID Connectは、**Google、Facebook、LINE、Yahoo! JAPAN、AppleなどのIDプロバイダーによって採用**されています。これらの大手プラットフォームが対応していることから、**間接的に多くのユーザーがOpenID Connectを利用している**と推測できます。

企業での普及

- クラウドサービスの利用が企業でも一般化する中、新しいサービスへのログイン時にGoogleやFacebookなどの認証サーバーを使ったID連携が広まっています。これにより、企業ユーザーの間でもOpenID Connectの利用が増加していると考えられます。

技術的な採用

- **多くのWebアプリケーションやモバイルアプリが、ユーザー認証にOpenID Connectを利用**しています。これにより、ユーザーは新規登録の手間を省き、既存のアカウントでログインできるようになっています。

将来的な普及

- **OpenID Connectは、業界大手が仕様策定作業に参画しており、実装も積極的になされているため、今後さらに普及する**ことが予想されています。
- ただし、具体的な利用者数を示す統計データは見つかりませんでした。OpenID Connectは認証の仕組みであり、多くのユーザーが意識せずに利用している可能性が高いため、正確な利用者数の把握は難しいと考えられます。

What We Achieved

- Most used identity protocol
- Thousands of interoperable implementations
  - In every conceivable language
- Certification Program making interop a reality
- ISO accepted our submission for republication

SAML or OpenID Connect by Perplexity

用途による選択

- 企業向けシステム
  - **SAMLは企業システム間の認証に広く利用されており**、社内の複数アプリケーションへのシングルサインオンに適しています。
- 消費者向けサービス
  - **OpenID Connectは消費者向けWebサイトやモバイルアプリでの利用に適しており**、ソーシャルログインなどに広く採用されています。

技術的特徴

- SAML
  - XMLベースのプロトコル、**仕様が複雑で実装が難しい**、企業システムとの親和性が高い
- OpenID Connect
  - JSONベースのプロトコル、**実装が比較的容易**、モバイルアプリやWebアプリとの相性が良い

まとめ

- **企業内システムの認証連携にはSAMLを、消費者向けサービスにはOpenID Connectを採用するのが一般的**です。ただし、具体的な要件や連携先システムの状況を踏まえて判断することが重要です。両方の規格に対応できるソリューションを選択することで、柔軟な対応が可能になります。

学術機関におけるIdP/SPの傾向（私的観測）

- **IdP**
  - Shibboleth一択からEntra ID（Microsoft）やExtic（Exgen）などの**IDaaSへの切り替えが進んできた**
    - Shibbolethのバージョンライフサイクルの問題、破壊的変更の多さ
    - サーバインフラの運用の手間とコスト
  - **IDaaSのプライマリ対応プロトコルはOpenID Connectにシフト**しつつある
- **SP**
  - Shibboleth SPを組み込む自前アプリから**SaaSの利用拡大**が進んできた
    - BoxやMicrosoft 365など
  - まだエンタープライズ向けのSPはSAML対応しているものもあるが、**OpenID Connect対応のものが増えてきた**

現実論としての機関IdP

- **SPファースト**で機能選択が行われる
  - 繋ぎたいSPが対応しているプロトコルが重要
  - キラーアプリは？
- **学認対応のためのSAML/Shibboleth？**
  - 学内利用はEntra ID、学認連携用にShibboleth、という多段構成にすることも

SAMLを知っている人向けのOpenID Connect

やれることは表層的には殆ど変わらない

- **ID連携（Federation）のためのプロトコル**
  - **IdPで発行されたアサーションをSPへ連携する**仕組み
  - 結果、認証情報の集中管理やSSOによる利便性提供を実現
- ものすごく雑にいうと
  - **SAMLはXML、OpenID ConnectはJSON**でID情報を表現する
  - **ID情報のやり取りはどちらもHTTPベースで実装が可能**なのでほぼ変わらず
  - **バインディングの種別も類似**（主流は若干異なるが同じことはできる）
    - SAMLの主流はHTTP POST Binding（OpenID Connectでいうところのimplicit flow）
    - OpenID Connectの主流はcode flow（SAMLでいうところのArtifact Binding）

SAMLを知っている人向けのOpenID Connect

あえて比較してみると殆ど変わらない

	SAML	OpenID Connect
仕様策定時期	2005年（SAML2.0）	2014年（OpenID Connect 1.0）
ID情報（Assertion）のフォーマット	XML	JSON（JSON Web Token）
Assertionへの署名方法	XML Signature （正規化の上で署名）	JWS（JSON Web Signature） （正規化不要）
署名検証鍵の公開方法	Metadata交換	Metadata公開（JWKS_URI）
Assertionの暗号化方法	XML Encryption	JWE（JSON Web Encryption）
IdP-SP間の信頼関係	Metadata交換	Client登録
エンドポイント情報の探索	Metadata交換	Metadata公開（Discovery）
トランスポートプロトコル	HTTP、SOAP	HTTP
バインディング	フロントチャネル）HTTP POST、HTTP Redirect バックチャネル）HTTP Artifact ※HTTP系のみ（SOAP系は省略）	フロントチャネル）Implicit Flow バックチャネル）code flow

OpenID Connectの設計思想

- モジュラー型の設計
  - JWS/JWE/JWK/JWT/WebFinger/ID Tokenをうまく利用
- 最も重要な思想
  - making simple things simple and complicated things possible

  →シンプルなことは簡単に実装でき、複雑なことも実現できるように
- そのための意思決定
  - 正規化しない、ASCIIアーマー、JSON、REST
  - 全ては実装の簡素化によるバグ・脆弱性の入る余地を最小化するために

OpenID Connectの基本

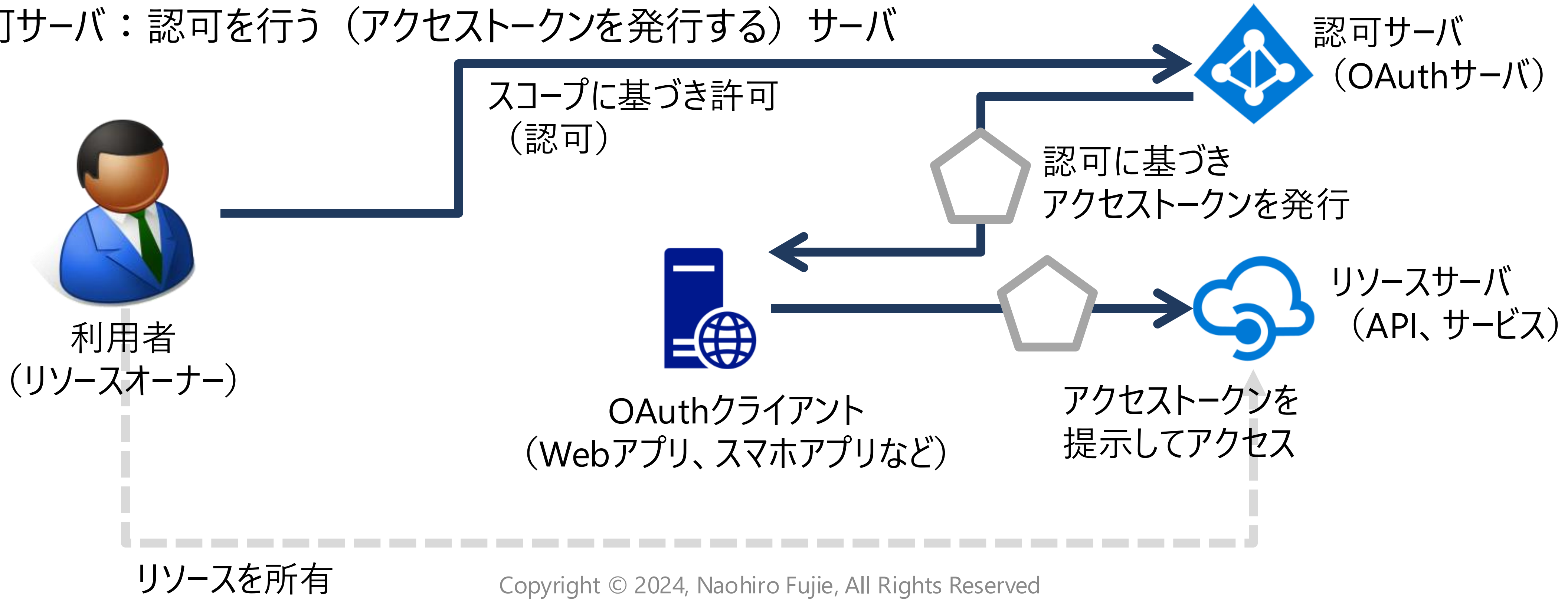
- OpenID Connect 1.0 は, OAuth 2.0 プロトコルの上にシンプルなアイデンティティレイヤーを付与したものである. このプロトコルは Client が Authorization Server の認証結果に基づいて End-User のアイデンティティを検証可能にする. また同時に End-User の必要最低限のプロフィール情報を, 相互運用可能かつ RESTful な形で取得することも可能にする.
- OpenID Connect Core 1.0 日本語訳
  - http://openid-foundation-japan.github.io/openid-connect-core-1_0.ja.html
  - OpenIDファウンデーション・ジャパン翻訳・教育WG

OAuth2.0とは

- OAuth 2.0 は, サードパーティーアプリケーションによるHTTPサービスへの限定的なアクセスを可能にする認可フレームワークである.
- RFC 6749: The OAuth 2.0 Authorization Framework 日本語訳
  - http://openid-foundation-japan.github.io/rfc6749.ja.html
  - OpenIDファウンデーション・ジャパン翻訳・教育WG

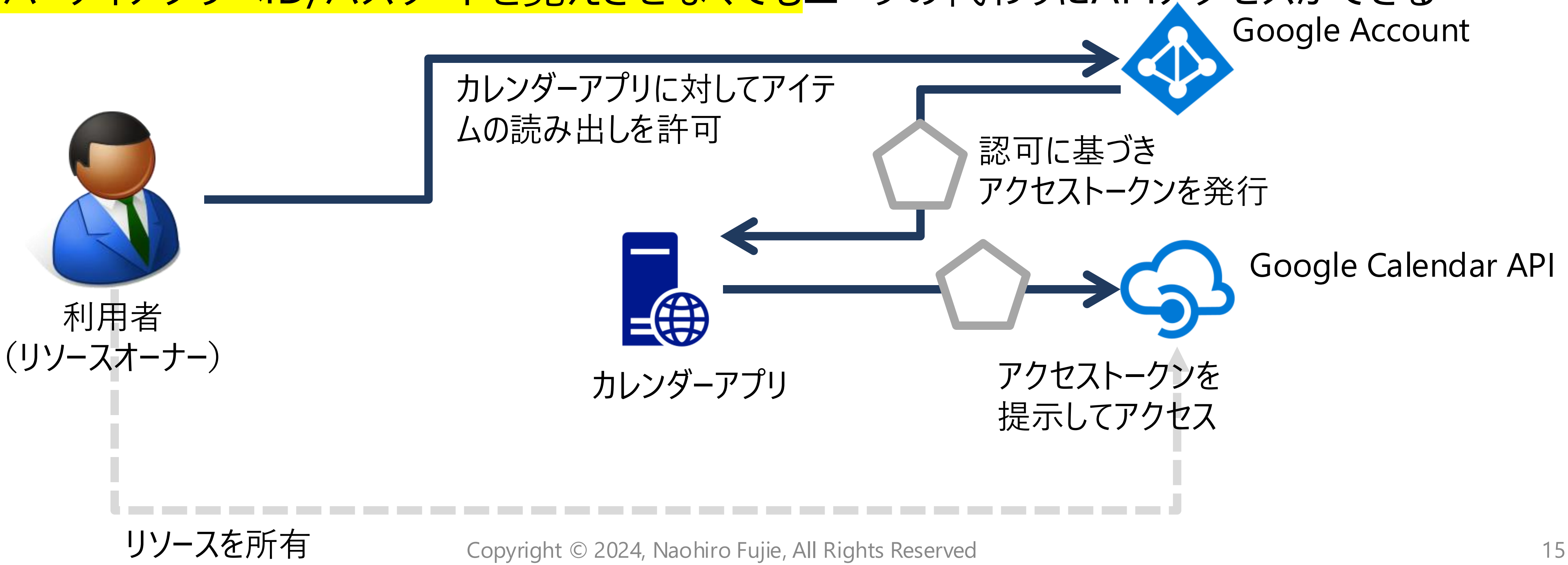
基本的な構成要素と考え方（OAuth2.0）

- リソースオーナー：利用者（リソースサーバで提供されるリソースの持ち主）
- OAuthクライアント：Webアプリ、スマホアプリなど
- リソースサーバ：API、サービスを提供するサーバ
- スコープ：認可の範囲を表す
- 認可サーバ：認可を行う（アクセストークンを発行する）サーバ



典型的な使われ方

- 3rdパーティのカレンダーアプリからGoogleカレンダーへアクセスする許可を与える（利用者に代わってカレンダーアプリがGoogleカレンダーへアクセスすることを認可する）
- アクセス制御というよりも「権限委譲」という意味合いの方が近い
- 3rdパーティアプリへID/パスワードを覚えさせなくてもユーザの代わりにAPIアクセスができる



IDのレイヤーを付加する→OpenID Connect

- OAuthクライアントがリソースサーバのID情報を取得する
  - リソースオーナーの認証結果、属性情報など
- OpenID Connectでの呼び方
  - OAuthクライアント＝RP/Relying Party（SAMLでいうSP）
  - リソースサーバ＝OP/OpenID Provider（SAMLでいうIdP）
- ID情報の取得
  - スコープとしてopenidを指定することでアクセストークンに加えてid_token（SAMLでいうAssertion）を取得
  - スコープとしてemailやprofileなどを指定することでuserInfoエンドポイント（通常IdPが提供するAPIエンドポイント）からID情報（属性など）を取得

※id_tokenにもID情報を含められるがトークンのサイズの巨大化の問題や認証時以外の情報取得に対応するためuserInfoエンドポイントを併用するケースが多い

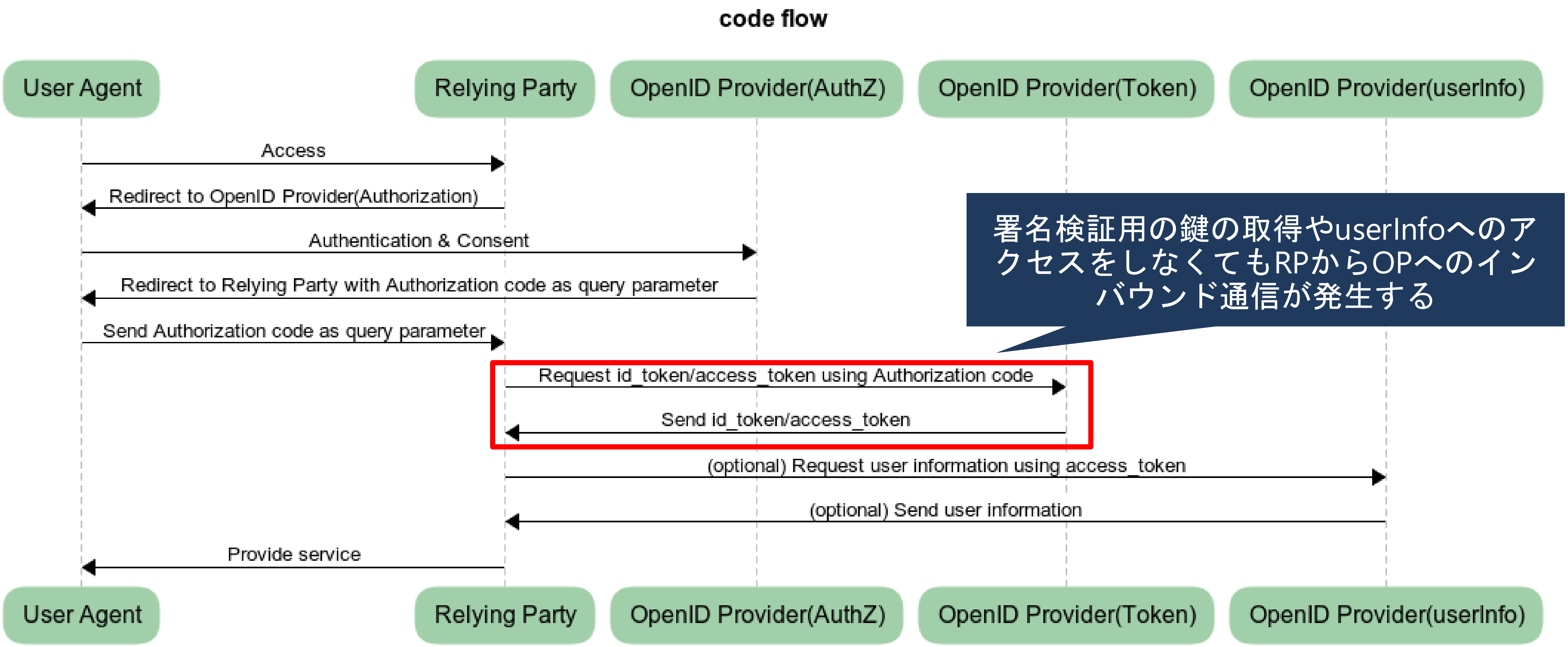
→ID連携のいつもの動き（RPへアクセスするとOPへリダイレクト、認証結果とユーザ属性が返却される）へ

基本的なフロー

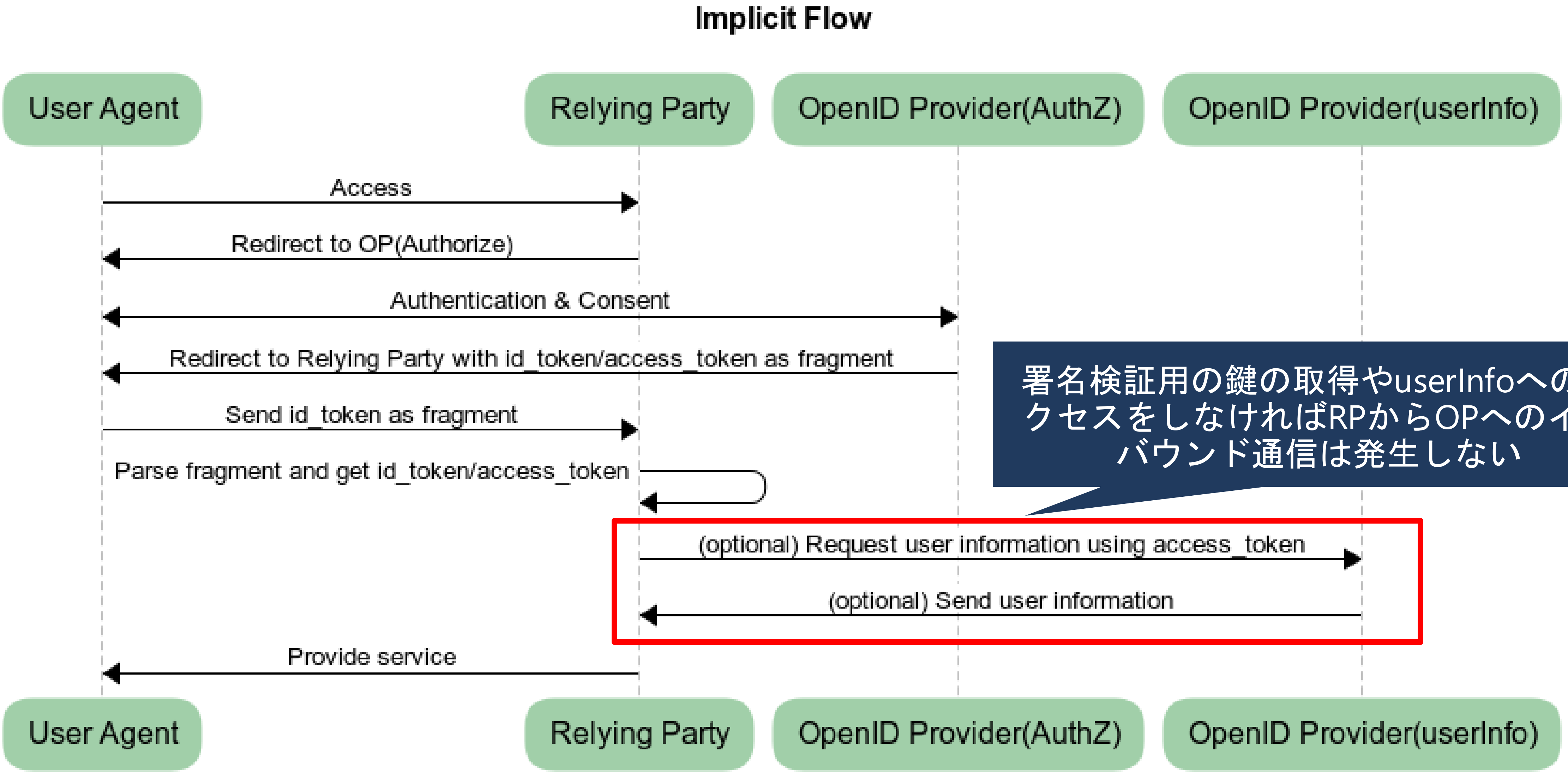
- SAMLでいうところのBinding（雑な説明）
- 以下の3つが定義されている

フロー	特徴	RPからOPへの直接通信	SAMLでいうところの
Code flow	一番基本的なフロー バックエンドで処理	あり	Artifact Binding
Implicit flow	ファイアウォールなどでRPからOPへ直接通信できない環境などで利用するためフロントエンドで処理	なし ※jwks_uri, userInfoを使わない場合	HTTP Redirect Binding
Hybrid flow	Code flowをよりセキュアにするためのフロー	あり	該当なし

Code flow


code flow
User Agent
Relying Party
OpenID Provider(AuthZ)
OpenID Provider(Token)
OpenID Provider(userInfo)
Access
Redirect to OpenID Provider(Authorization)
Authentication & Consent
Redirect to Relying Party with Authorization code as query parameter
Send Authorization code as query parameter
Request id_token/access_token using Authorization code
Send id_token/access_token
(optional) Request user information using access_token
(optional) Send user information
Provide service
User Agent
Relying Party
OpenID Provider(AuthZ)
OpenID Provider(Token)
OpenID Provider(userInfo)
署名検証用の鍵の取得やuserInfoへのアクセスをしなくてもRPからOPへのインバウンド通信が発生する

Implicit flow



署名検証用の鍵の取得やuserInfoへのアクセスをしなければRPからOPへのインバウンド通信は発生しない

id_token（SAML Assertionに相当）

- JWT（JSON Web Token）形式
  - JSON Web Token (JWT) - draft-ietf-oauth-json-web-token-11 日本語訳
  - http://openid-foundation-japan.github.io/draft-ietf-oauth-json-web-token-11.ja.html
  - OpenIDファウンデーション・ジャパン翻訳・教育WG
- 内部構造
  - ヘッダ：署名や暗号化形式など
  - ペイロード：クレーム（属性）セット、暗号化する場合も
  - シグニチャ：デジタル署名
- Base64Urlエンコードし、"."で各パートを連結する
  - eyJhb--snip--iJ9.eyJpc3--snip--pwIn0.gC4ub--snip--yBm0

id_tokenの中身（LINE OPの例）

Claim Type	Value	Notes
iss	https://access.line.me	JWTの発行者（issuer）を表す識別子
sub	U9f1cac4f164ef3f5c02c92d0067a11a1	JWTの主体（subject）を表す識別子 LINEの場合はuserId
aud	1516319320	JWTの発行先（audience）を表す識別子 LINEの場合はclient_id
exp	1552324580	JWTの有効期限（UNIX Time）
iat	1552320980	JWTの発行時刻（UNIX Time）
nonce	51501f6a-9a12-4d42-ad72-0d36e44df96f	リクエスト時に設定したnonceの値 リクエストと発行されたid_tokenの中の値がマッチするかどうかを検査し置き換え攻撃を検知する
name	Naohiro Fujie	名前。LINEの場合は表示名
picture	https://profile.line-scdn.net/0m0--snip-- xxx	プロファイル写真のURL
email	naohiro.fujie@xxxxxxx.jp	メールアドレス

OpenID Connect/OAuthの応用

- OAuthはフレームワーク
- OpenID Connectを含め様々な仕様やプロファイル（ユースケースに特化した使い方）を策定
- 例）
  - 身元確認）OpenID Connect for Identity Assurance
  - 高セキュリティ環境での利用）Financial-grade API Security Profile
  - トラスト確立）OpenID Federation
  - 資格情報の発行・提示）OpenID for Verifiable Credentials関連仕様
  - リスク情報の共有）Shared Signals Framework

身元確認とトラストフレームワーク

- OpenID Connect for Identity AssuranceではOPによるアイデンティティに関する保証状態を表現可能
  - 例）
    - XXXというトラストフレームワークに基づき
    - YYYという確認書類を使い
    - 対面で
    - トランザクション番号ZZZで
    - 確認を行った
- 組織と個人の関係性の表現についても策定中
  - 個人が組織の中でどのような権限を持つか、誰がその権限を付与したか、など

OpenID Connect for Identity Assuranceの例

"verified_claims": {
"verification": {
"trust_framework": "authority_claims_example_framework",
"time":"2020-04-23T18:25Z",
"verification_process":"f24c6f-6d3f-4ec5-973e-b0d8506f3bc7"
},
"claims": {
"given_name": "Bob",
"family_name": "Smith",
"birthdate": "1981-01-26",
"authority": [{
"applies_to": {
"organization_name": "Example Company Limited",
"trading_as": "XAmple",
"registered_address": "123 Acacia Avenue, Newtown, UK",
"registration_number": "12351235",
"registration_authority_code": "RA000585",

身元確認がどのトラストフレームワークに則っていつ実行されたのか？

確認済みの属性情報

確認済みのアイデンティティ関連する組織の情報

機関IdPのOpenID Connect対応時の留意点

- 色々と選択肢が出始めてきている
  - Shibboleth or IDaaS or オンプレIdP
    - ShibbolethはいつまでたってもPlugin扱い
- 留意点（結局はどんなSP/RPを使うか次第）
  - IdPの配置とフローをどうするか（ファイアーウォールの中？外？IDaaSかどうか）
    - インバウンド通信を許可したくなければImplicit flowとなるがjwks_uriやuserInfoをどうできるかはRP次第（RP側にスタティックに公開鍵を設定できるかどうか、userInfoのアクセスを止められるかどうか）
  - id_tokenを暗号化するか（意外と対応していないOP＆RPが多い）
    - 学認IdPのノリでカジュアルに考えているとダメなこともある
  - 学認IdPと兼用にするか（SAMLが喋れるか、学認メタデータを読み込めるか）
    - SAMLは喋れても学認メタデータを読み込むところは開発が必要（開発できる余地があればまだマシ）
    - 結局Shibbolethと多段構成にするとIDaaSのメリットは激減

その他留意点）認証コンテキストの話

- 認証コンテキストと認証手段を表すClaim
  - SAML：AuthnContextClassRefでまとめて表現
  - OpenID Connect：acrとamrに分かれている
- SPからの認証コンテキストの要求
  - SAML：AuthnRequest内でAuthnContextClassRefを指定
  - OpenID Connect：リクエスト時にacr_valuesをqueryで指定

→結構対応していないOP/RPが多いので注意

（特に次世代学認に対応させようとすると注意）

その他留意点）識別子の話

- 学認の識別子フォーマット（学認運用基準より）
  - nameid
    - urn:oasis:names:tc:SAML:2.0:nameid-format:transient（匿名値）
  - eduPersonTargetedId
    - urn:oasis:names:tc:SAML:2.0:nameid-format:persistent（仮名値）
- OpenID Connectにおける識別子（sub）
  - Subject Identifier Type
    - public：すべてのRPへ同一の値を提供する
    - Pairwise：RPごとに一意の値を提供する（仮名）
  - SAMLのtransientに該当するフォーマットは存在しない

その他の話題

- SAML is Deadの真の意味
  - 十分に成熟、仕様拡張はほぼされない→良い意味で枯れた技術
- OpenID Connect/OAuth周辺仕様は進化中
  - Internet Identity WorkshopではAuthnContextClassRefをSAML/OpenID Connectで共通化する取り組み（by Pam Dingle）
  - エンタープライズ向けアイデンティティに関する新WG「Interoperability Profiling for Secure Identity in the Enterprise」の設立に向けた動き
  - 資格証明（Verifiable Credentials関係）の利活用を見据えた仕様
  - オープンエコシステムの中で仕様開発（ISO/IEC、W3C、IETF、DIF等と協業）

学術機関における次世代IdPの
あるべき姿を一緒に考えていきましょう！