# 認証方式とSSOの基礎知識

2025年7月4日

# 改定履歴

| 版数 | 発行日 | 改訂内容 |
|---|---|---|
| 第1版 | 2025年7月4日 | 初版発行 |
| | | |

本資料の内容は 2025/7/4 時点のものです。製品のアップデートにより変更となる場合がございます旨でご了承ください。

# Agenda

1. 前提情報
    1. 本書の目的とゴール
    2. 用語集
2. 認証方式の概要
    1. 認証・認可とは
    2. IDとパスワードによる認証の限界
    3. SSO（シングルサインオン）の必要性
3. 主要な認証プロトコル
    1. 主要な認証プロトコル
    2. SAML
    3. OIDC
    4. SAMLとOIDCの比較

# 1. 前提情報

# 1.1. 本書の目的とゴール

## 目的

クラウドサービスの利用やシステム連携において求められる**認証の基礎を理解**し、認証方式に関する知識の向上を図ることを目的とします。本資料では、主要な認証プロトコルや構成要素、認証フローについて、基礎から体系的に学習できる内容としています。

## ゴール

本資料を学ぶことで、以下の内容を理解し、認証方式の選定や設定において基本的な判断・検討ができる状態を目指します。

1. **認証と認可の違い、基本概念**
2. **SSOの必要性とID・パスワード認証の課題**
3. **SAML / OpenID Connect の特徴と違い**
4. **各認証方式における構成要素・認証フロー**

# 1.2. 用語集

本書で使用する用語及び略称を以下の通り定義します。

| No. | 用語 | 説明 |
|---|---|---|
| 1 | PINコード | ATMやスマートフォンなどで本人確認に使われる短い数字列。セキュリティ強化のため、他の認証手段と組み合わせることも多い。 |
| 2 | トークン | 認証・認可のために発行される一時的な文字列。アクセストークンやIDトークンなどがあり、ユーザーの認証状態や権限を示す。 |
| 3 | TOTP<br>（Time-based One-Time Password） | 時間ベースで生成される使い捨てパスワード。一定時間ごとに新しいコードが生成され、スマホアプリ（例：Microsoft Authenticator）で利用される。 |
| 4 | OTP（One-Time Password） | 一度限り有効なパスワード。ログイン時の追加認証としてSMSやメールで送信されることが多く、パスワード漏洩対策に有効。 |
| 5 | キーロガー | ユーザーのキーボード入力を密かに記録するソフトウェア。マルウェアの一種で、IDやパスワードなどの情報を盗む目的で使われる。 |
| 6 | マルウェア | 「Malicious Software」の略。ウイルス、スパイウェア、ランサムウェアなど、悪意のある動作をするソフトウェアの総称。 |
| 7 | FIDO2 / WebAuthn | パスワード不要の認証を実現する標準技術。指紋認証やセキュリティキー（YubiKeyなど）を使い、フィッシング耐性が高い。 |
| 8 | Cookie | Webサイトがユーザーのブラウザに保存する小さなデータ。ログイン状態の保持やユーザー追跡、パーソナライズに利用される。 |
| 9 | XML | データ構造を記述するためのマークアップ言語で、主に古いWebサービスや設定ファイルで使用される。 |

# 1.2. 用語集

本書で使用する用語及び略称を以下の通り定義します。

| No. | 用語 | 説明 |
|---|---|---|
| 10 | SOAP<br>（Simple Object Access Protocol） | XMLを使ったWebサービス通信プロトコル。堅牢だが、RESTよりも複雑で重い。<br>REST：Webサービスの設計スタイルの一つで、シンプルで統一された方法でクライアントとサーバーが通信できるようにするアーキテクチャ原則。 |
| 11 | HTTP | 「HyperText Transfer Protocol」。Webブラウザとサーバー間の通信に使われる基本プロトコル。HTTPSは暗号化されたバージョン。 |
| 12 | JSON | 軽量で人間にも読みやすいデータ形式。REST APIなどで広く使われている。 |
| 13 | JWT（JSON Web Token） | 認証情報を含むトークン形式。署名付きで改ざん防止が可能。IDトークンやアクセストークンとして使われる。 |
| 14 | SPA（Single Page Application） | ページ遷移なしで動作するWebアプリ。ReactやVueなどのフレームワークで構築され、ユーザー体験がスムーズ。 |
| 15 | フェールオーバー構成 | システム障害時に自動で予備系に切り替える構成。高可用性を実現し、サービスの継続性を保つ。 |
| 16 | デジタル署名 | 電子的に文書の正当性と改ざんの有無を保証する技術。公開鍵暗号を使い、本人性と整合性を担保する。 |
| 17 | エンタープライズSSO | 企業内の複数の業務システムに対して、1回のログインでアクセスできる仕組み。ユーザーの利便性とセキュリティを両立する。 |
| 18 | レガシーシステム | 古い技術や設計で構築され、現在も業務で使われ続けているシステム。保守が困難で、他のシステムとの連携や拡張が難しいことが多い。 |
| 19 | Confidential Client | クライアントシークレットを安全に保持できるアプリケーションで、主にサーバーサイドのWebアプリなどが該当。<br>OAuth 2.0では、これらのクライアントが認可コードフローなどの安全な認証方式を利用して、ユーザーのデータにアクセスする。 |

# 2. 認証方式の概要

# 2.1. 認証・認可とは

本章では、まず認証の基本的な概念と認可との違いを明確にします。

**認証とは**

通信の相手が「誰（何）であるのか」を**利用者本人か確認・特定すること**です。
正しいユーザーだけがシステムやサービスにアクセスできるようにするため、セキュリティの第一歩として非常に重要です。

認証ではユーザーを特定するための方法として、次の3要素のいずれか、またはそれらを組み合わせて確認を行います。
これらの要素を基に、多要素認証や生体認証など、様々な認証の種類が実際に使われています。

**知識情報**
**(Something You Know)**
パスワード、PINコード、秘密の質問など

**所持情報**
**(Something You Have)**
IDカード、トークン、スマホなど

**生体情報**
**(Something You Are)**
指紋、顔、虹彩、声紋などの生体情報

| 認証の種類 | 説明 | 使用例 |
|---|---|---|
| **パスワードベースの認証** | 最も基本的な認証方式。ID＋パスワードで本人を確認。 | Webサービスのログイン |
| **証明書ベースの認証** | 暗号化されたデジタル証明書を使って、デバイスやユーザーの本人確認を行う方法。 | マイナンバーカード、VPN接続 |
| **生体認証** | 指紋や顔、網膜など生体情報を使う認証。パスワードより安全でユーザビリティに優れる。 | スマホの指紋認証や顔認証 |
| **トークンベースの認証** | TOTPやOTP（ワンタイムパスワード）など、一時的なコードで本人確認する認証方式。 | Microsoft Authenticatorアプリ |
| **プッシュ通知認証** | ログイン時にスマホへ通知を送り、承認ボタンで認証する方式。OTPと組み合わせることもある。 | Microsoft Authenticatorのプッシュ通知 |
| **音声認証** | 音声通話などを通じて本人確認を行う方式。 | コールセンターでの本人確認 |
| **多要素認証（MFA）** | 上記の複数方式を組み合わせ、2つ以上の認証要素を用いて強固な認証を実現。 | ①ID+PW ②スマホのTOTPコード |

# 2.1. 認証・認可とは

**認可とは**

認証されたユーザーに対して**「どのリソースに、どの範囲までアクセスを許可するか」を判断・制御する仕組み**です。
これにより、システム内の情報漏洩や不正操作を防ぎ、適切なアクセス管理を実現します。

**認可で可能となること**

1. **アクセス可能な端末の限定**
   会社支給の端末のみアクセスを許可するなど、利用端末を制限することでセキュリティリスクを低減します。
   私用端末からのアクセスを防ぎ、情報漏洩や不正アクセスを防止します。
2. **アクセス可能な場所の限定**
   特定のIPアドレス範囲以外からのサービスアクセスを制限するなど、アクセス元を制限することで不正アクセスを防ぎます。
3. **必要な権限の付与**
   ユーザーの役割や属性に応じてアクセス可能な範囲（例：閲覧、編集、承認）を細かく設定し、適切な情報管理を実現します。

**認可の代表的な方法：ロールベースアクセス制御（RBAC：Role-Based Access Control）**

「役割（ロール）」にドキュメントへの権限を付与し、ユーザーをその役割に割り当てることで権限を管理します。

- 管理職（部長、課長）：重要文書や一般文書の閲覧、承認、編集、共有が可能。
- 一般社員：一般文書の閲覧、一部編集が可能。重要文書の閲覧は不可。
- 監査担当者： 特定の部署や全社のドキュメントを閲覧・確認のみ（変更・削除不可）。

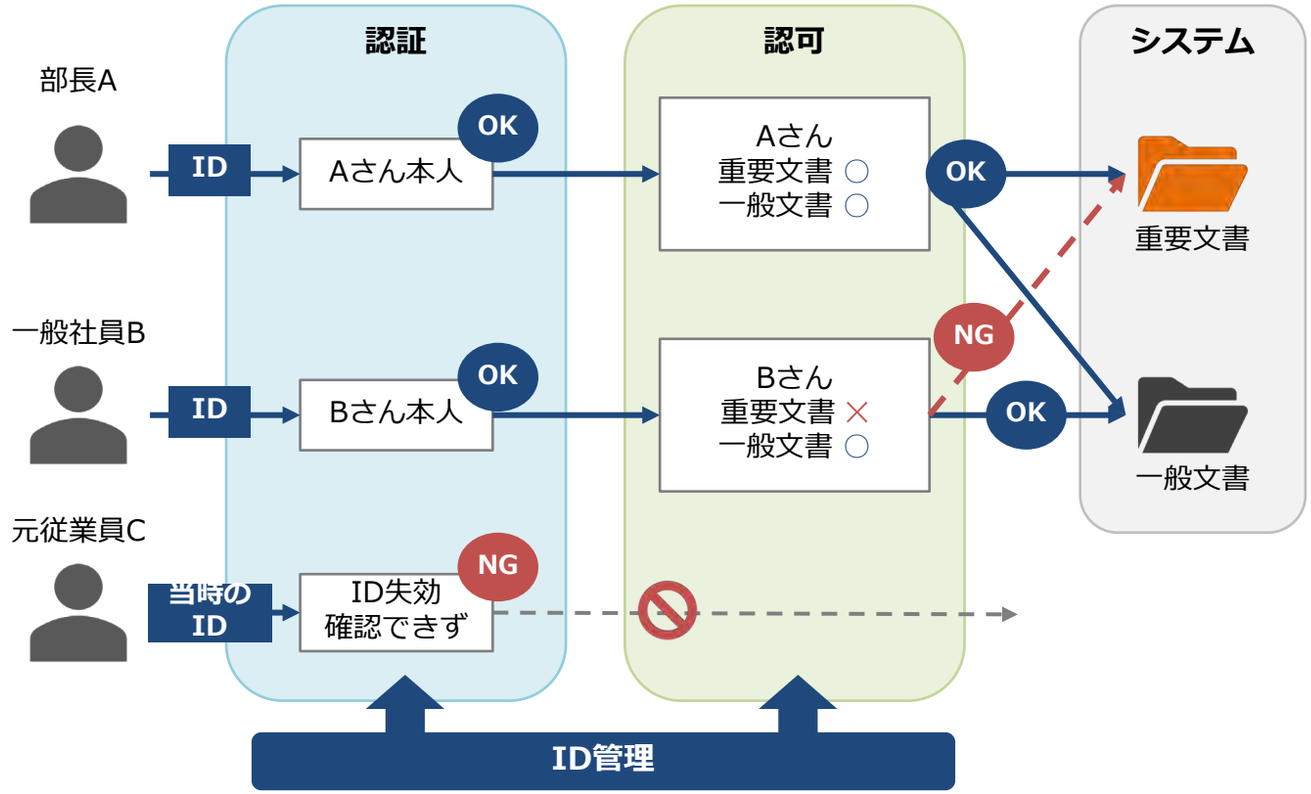
# 2.1. 認証・認可とは

認証と認可は、システムやデータへのアクセスを安全に管理するために、それぞれ異なる役割を担いながら連携する重要な仕組みです。

本人確認 ＝ 認証　　アクセス制御 ＝ 認可

## 認証と認可の連携とフロー

1. **本人識別**：ユーザー名やIDを入力し、ユーザーが自身の身元を名乗る。
2. **認証**：パスワードやワンタイムパスワード、生体情報などで**本人であることを確認**する。
3. **認可**：認証済みユーザーが、**どのリソースにどの範囲までアクセスできるかを制御**する。




認証と認可をうまく組み合わせることによって、正しい本人確認と、正しいアクセス制御が実現でき、情報に対して適切な扱いを実現できます。

# 2.2. IDとパスワードによる認証の限界

認証がデジタルサービスの入口を守る重要な役割を担う一方で、その中心にあるIDとパスワード認証だけでは、現代の複雑な脅威からユーザーやシステムを守りきれないという限界が指摘されています。

## ID/パスワード認証の限界

### ユーザー側の課題

- **パスワードの使い回し**
  情報漏洩時、他サービスへの不正アクセスリスクを高める。
- **脆弱なパスワード**
  推測されやすく、攻撃により容易に解読される。
- **管理負担**
  多数のパスワード記憶・管理はユーザーに大きな負担となる。

### 攻撃側の進化

- **パスワードリスト攻撃**
  漏洩した情報を利用し、他サービスへのログインを試みる。
- **フィッシング詐欺**
  偽サイトやメールでユーザーを騙し、認証情報を詐取する。
- **キーロガー・マルウェア**
  デバイスに潜伏し、入力情報や認証情報を盗み取る。

## 対策：パスワードに依存しない認証

### 技術的対策

- **多要素認証 (MFA)**
  異なる認証要素を複数組み合わせ認証の層を厚くする。
  例: SMS、認証アプリ、生体認証、セキュリティキー
- **パスワードレス認証**
  パスワード自体を使用しないため、ユーザー負担とフィッシングリスクを解消。例: FIDO2/WebAuthn
- **シングルサインオン (SSO)**
  一度の認証で複数のサービスにアクセス可能にする仕組み。
  セキュリティ対策（MFAやパスワードポリシーの強化など）と併用することで、強固な認証基盤を構築できる。

### 意識・運用による対策

**ユーザーとシステム双方の意識向上**

- ユーザー：複雑なパスワード設定、フィッシング詐欺への警戒。
- システム：MFAの強制、SSOの導入、不正ログイン検知。

# 2.3. SSO（シングルサインオン）の必要性

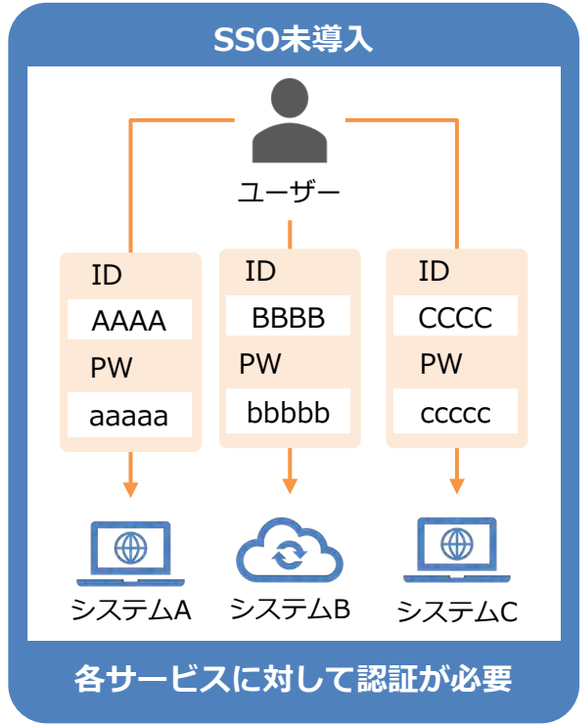
ID/パスワード認証の限界に加え、Web・クラウドサービス普及に伴うシステム増加は、ユーザーにセキュリティリスクと利便性低下の課題をもたらしています。これらの解決と安全性・利便性の両立には、SSO（シングルサインオン）が不可欠です。

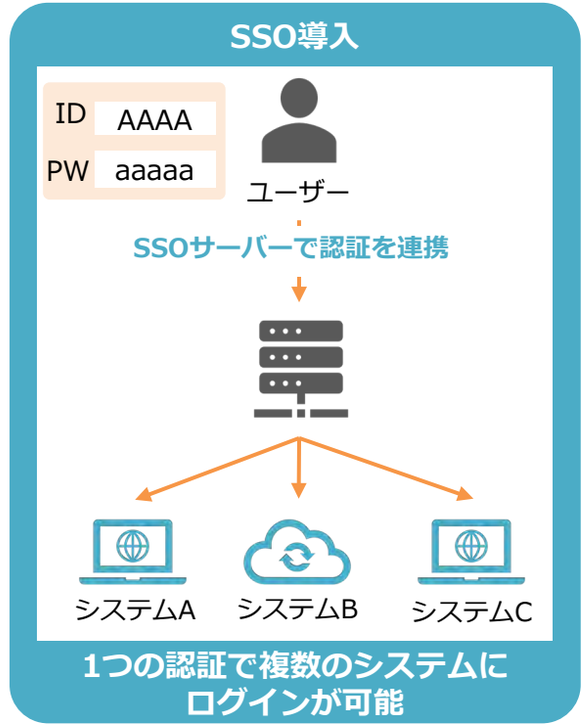
## SSO（Single Sign-On）とは？

ユーザーが**一度認証するだけで、複数のサービスに安全にアクセスできる仕組み**です。
個々のサービスごとにIDとパスワードを入力する手間が不要になり、利便性とセキュリティを両立できます。

## SSOの必要性

1. **利便性向上**
   一度の認証で複数システムにアクセス可能。
   ユーザー側のパスワード管理の手間を大幅削減。
2. **セキュリティ強化**
   一元的な認証により、脆弱なパスワードや使い回しを防ぎ、パスワードを盗まれるリスク（フィッシングなど）も低減。
   認証の一元管理で、迅速な不正アクセス対応が可能。
3. **管理負担軽減**
   パスワードリセットやアカウント管理、権限変更の効率化。
4. **コンプライアンス対応**
   アクセス履歴の一元管理により、監査・ポリシー適用が容易。



# 2.3. SSO（シングルサインオン）の必要性

## SSOの認証方式の種類

SSOを実現するには5つの方式があります。

| 種類 | 説明 |
| --- | --- |
| エージェント方式 | Webシステムやアプリケーションのサーバにエージェントモジュールを組み込み、認証サーバと連携して認証情報を管理する方式。既存ネットワーク環境で導入しやすく、拡張性が高い点がメリット。全サーバへの導入が必要で、レガシーシステムには非対応な場合があるのがデメリット。 |
| リバースプロキシ方式 | クライアントとサービスの間に中継サーバ（リバースプロキシ）を設置し、通信を制御・認証する方式。認証後にCookieを発行してアクセスを許可する。専用エージェントが不要で導入が容易だが、ネットワーク構成の最適化が必要。 |
| 代理認証方式 | 端末にエージェントを導入し、ログイン画面を検知して認証情報を自動入力することでSSOを実現する方式。レガシーシステムにも対応でき、サーバ側の改修が不要なのが利点。一方で、エージェントの導入が必須であり、ID情報の整合性管理が課題となる。 |
| **フェデレーション方式** | 異なる組織やドメイン間で、**共通の認証プロトコル（SAMLやOpenID Connectなど）を使い、SSOを実現する**方式。<br>クラウドサービスとのSSOに適しており、ユーザー属性に基づく柔軟な認可制御も可能。対象のシステムがプロトコルに対応していれば、比較的容易にSSOを構成できるが、非対応の既存システムでは、改修や追加コンポーネントが必要になる場合がある。 |
| 透過型方式 | 端末とWebシステムの間に監視用のSSO製品を導入し、通信を監視して必要時のみ認証情報を送信する方式。エージェントのインストールやネットワーク構成の変更が不要で、比較的新しい方式とされる。 |

これらの方式は、それぞれメリット・デメリットや適した利用シーンが異なります。

クラウドサービスの利用が拡大し、企業内外の多種多様なサービス連携が不可欠な現代において、高いセキュリティと相互運用性を持つ**フェデレーション方式がSSOの主流**となります。

今回は、このフェデレーション方式の**SAML**や**OpenID Connect (OIDC)** について次の章から詳しく解説します。

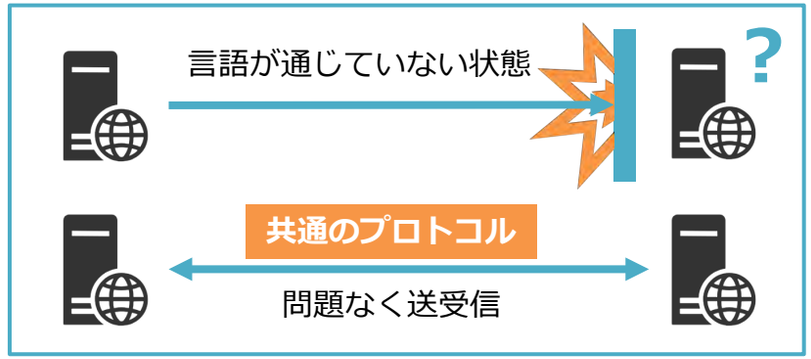
# 3. 主要な認証プロトコル

# 3.1. 主要な認証プロトコル

本章では現代の認証・認可を支える主要なプロトコルについて詳しく解説します。
まずは『プロトコル』の基本概念から確認し、SAML、OpenID Connect (OIDC)の概要を説明します。

**プロトコルとは？**

- コンピュータやデバイス間で情報をやり取りするための、**共通の「約束事」や「手順」**。
- 異なるベンダーやプラットフォーム間でも、共通のプロトコルを用いることで、データの送受信やスムーズな連携が可能となり、ユーザーの身元確認やアクセス権の付与といった認証・認可を安全かつ確実に行えるようになる。




本章では特に利用頻度が高く重要な、以下2つの認証プロトコルについて詳しく解説します。
SAMLとOpenID Connectは、どちらもユーザーの認証とリソースへの認可を可能にしますが、それぞれ異なる得意分野と特性を持っています。
次以降のスライドで、それぞれの特徴や認証フローを見ていきます。

| 比較項目 | SAML | OpenID Connect (OIDC) |
|---|---|---|
| **技術基盤** | XMLベースで、SOAPやHTTPなどを使用 | OAuth 2.0をベースとし、JSON (JWT) を使用 |
| **主な用途** | エンタープライズSSOやクラウドサービスとの認証連携 | WebサービスやモバイルアプリのSSO、APIの認証 |
| **提供情報** | ユーザーの認証情報（属性情報含む）と認可情報（ロールなど）をXML形式のアサーションで提供。 | ユーザーの認証情報（IDトークン：JSON形式）と、OAuth 2.0によるリソースアクセス認可（アクセストークン）を提供。 |
| **導入の柔軟性** | 実装が複雑、モバイルに不向き | 軽量で柔軟、モバイルアプリやSPAと親和性が高い |

# 3.2. SAML

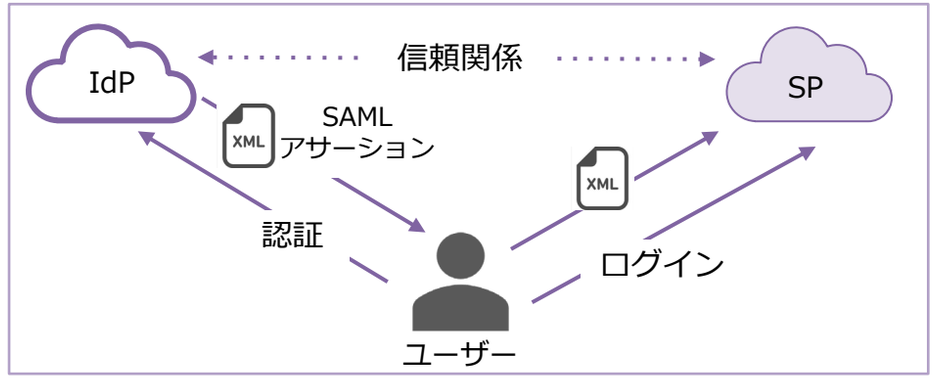
# 3.2.1. SAMLとは

## SAML（Security Assertion Markup Language）とは

**異なるドメイン間でユーザーの認証情報や認可情報を安全にやり取りするための、XMLベースの標準プロトコル**です。
社内のIdPと連携し、一度ログインするだけで複数のサービスを利用できる**SSOを実現**します。

## 特徴

- **IdP（Identity Provider）とSP（Service Provider）間の認証連携**
  SAMLは、**認証を行うIdP**（Entra IDなど）と、**サービスを提供するSP**（Boxなど）が連携して動作します。
  ユーザーがIdPで認証されると、その認証結果がSPに送信され、ユーザーはパスワードを再入力することなくSPにログインできます（SSOを実現）。
- **XMLベースの認証情報管理**
  認証や認可の情報を、**構造化されたXML形式（SAMLアサーション）でやり取り**を行います。
  XMLにはデジタル署名を含めることができるため、改ざんの検出が可能であり、安全かつ信頼性の高い情報交換が実現できます。
- **認可情報（属性）とアクセス制御**
  IdPは、ユーザーの認証に加えて、属性情報（例：氏名、所属、役職、グループなど）をSAMLアサーションに含めてSPに送信します。SPはこの属性情報をもとに、
  ユーザーごとに**柔軟なアクセス制御や機能制限**を行うことができます。



# 3.2.1. SAMLとは

## 活用シーン

**✓ 企業内でのシングルサインオン（SSO）**

社員が一度のログインで複数の業務システムにアクセス

- 社員が朝、社内ポータルにログインすると、SAML認証を通じて勤怠管理、経費精算、グループウェアなど他のクラウドサービスにも自動的にログイン可能に。
- → 毎回ログインし直す必要がなく、業務効率が向上。

**✓ クラウドサービスの統合認証**

Microsoft 365、Salesforce、Boxなどとの連携

- 社内のIdP（例：Entra ID、Okta）を使ってSAML認証を構成し、Microsoft 365やSalesforceへのログインに使う。
- → 利用サービスが増えても、認証は一元管理可能。

## メリット

- **セキュリティ向上**
  パスワードの使い回しや、退職者などの放置アカウントによる不正アクセスを防止。
  認証を一元管理できるため、迅速なアカウント制御が可能。
- **業務効率化**
  一度のログインで複数サービスにアクセス（SSO）。
  ユーザーの利便性向上と、管理者の運用負荷軽減を実現。
- **クラウドサービスとの高い互換性**
  SAMLは標準プロトコルのため、Microsoft 365やSalesforceなど多くのクラウドサービスと容易に連携可能。

## デメリットと対策

- **IdPが停止すると全サービスにログイン不可**
  → 認証基盤が単一障害点となるため、可用性の高いIdPの採用やフェールオーバー構成の検討が重要。
- **SAML非対応サービスがある**
  → 一部のレガシーシステムや特殊なサービスはSAML非対応の場合も。
  → 可能な限り対応サービスに統一しつつ、リバースプロキシやID連携サービスなどで補完手段を整備。
- **認証情報が一元管理のため、漏洩した場合の被害が大きい**
  → 多要素認証（MFA）や端末認証、ログ監視、アクセス制限などのセキュリティ対策を併用することでリスクを軽減。

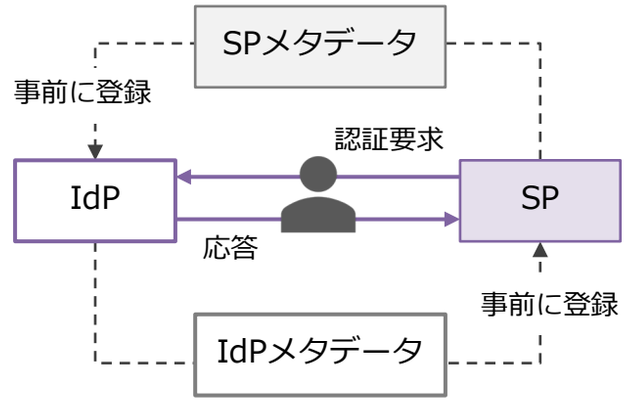
# 3.2.2. SAMLの構成要素

SAML認証は、ユーザー、認証を提供するサービス、利用するサービス、それらを連携するための情報から構成され、
これらの要素が次スライド以降で説明する認証フローの中で重要な役割を果たします。

| 構成要素 | 説明 |
|---|---|
| **IdP（Identity Provider）** | **ユーザーの認証情報や属性、権限を管理し、認証を行うシステム**。ユーザーのログイン情報はIdPが保持し、サービス提供者（SP）には認証結果を連携する。これにより、SPはパスワードを持たず、認証はすべてIdP側で行われる。<br>例：Microsoft Entra ID、Okta、Google Workspaceなど。 |
| **SP（Service Provider）** | **ユーザーにサービスを提供する側のシステム**。IdPで認証されたユーザー情報を受け取り、サービスを提供する。<br>IdPから受け取る認可情報を活用し、管理者機能の許可や特定ユーザーへの機能制限も行うことが可能。<br>例：Salesforce、Slack、Boxなど。 |
| **SAMLアサーション（Assertion）** | **IdPが発行するXML形式の認証情報**。ユーザーの認証状態や属性情報（名前、メールアドレスなど）、<br>認可情報（アクセス権限）、デジタル署名などの情報を持つ。 |

## SP と IdP の信頼関係

SAML認証において、**IdPとSPは相互に信頼できる相手**として事前に情報を登録し合うことが必要です。




**■ メタデータとは？**

- **宛先URL、証明書、エンドポイントなどの接続情報を記載したXML形式のファイル**です。
- 両者がこのメタデータを交換・登録することで、信頼関係が構築されます。

**■ なぜ信頼関係が必要か？**

- 信頼関係を結ぶことで SP は認証要求を、IdP は応答を送信できるようになります。
- SPはIdPから送信されたSAMLアサーションに含まれるデジタル署名を検証することで、その信頼性を確認します。

# 3.2.3. SAMLの認証フロー

SAML認証には、ユーザーがアクセスを開始する場所の違いによって「**SP起点**（Service Provider Initiated）」と
「**IdP起点**（Identity Provider Initiated）」の2つの認証パターンがあります。
このスライドでは、それぞれの違いを比較表で整理し、次のスライドから各認証フローの詳細を図を用いて解説します。

| 項目 | SP起点 | IdP起点 |
|---|---|---|
| **起点の違い** | **ユーザーが最初にSP（サービス側）にアクセス**し、<br>そこからIdPにリダイレクトされて認証が行われる。 | **ユーザーが最初にIdP（認証基盤側）にアクセス**し、<br>認証後にSPへのアクセス・ログインを行う。 |
| **ユーザーの操作** | SPにアクセス → IdPにリダイレクト → 認証 | IdPにログイン → SPを選択 |
| **ユースケース** | ブラウザブックマークや外部リンクなどから直接SPにアクセス<br>例）Boxにアクセス → Entra IDにリダイレクト<br>→ 認証 → Boxにログイン完了 | 企業ポータルサイトからの統合アクセス<br>例）Entra IDポータル（アプリリスト）にアクセス・認証<br>→ ポータルからBoxを選択 → Boxにログイン完了 |
| **メリット** | ユーザーが直接サービスにアクセスできる | ユーザーが一括管理されたポータルからサービスを選べる |
| **注意点** | SPが認証要求を正しく生成する必要がある | SP選択画面のユーザー体験やサービス連携の設定が重要 |

どちらのフローを選択するかは、主にユーザーが「**どこから**」認証を開始し、
「**どのように**」サービスを利用したいかというユーザー体験と、システムの管理・運用上の要件によって決まります。
多くのSSO環境では、両方のフローをサポートし、ユーザーの利便性や管理要件に合わせて選択できるように設計されています。

# 3.2.4. SAML認証のフロー（SP起点）

## SAML認証のフロー：SP起点

① ユーザーがWebブラウザから、**SPのサイトにアクセス**する。

② **SPはSAML認証要求を生成**する。
ユーザーのブラウザを**IdPへリダイレクト**させ、SAML認証要求をIdPに送信する。

③ IdPはユーザーに**ログイン画面を表示**し、認証情報（IDとパスワードなど）の入力を求める。

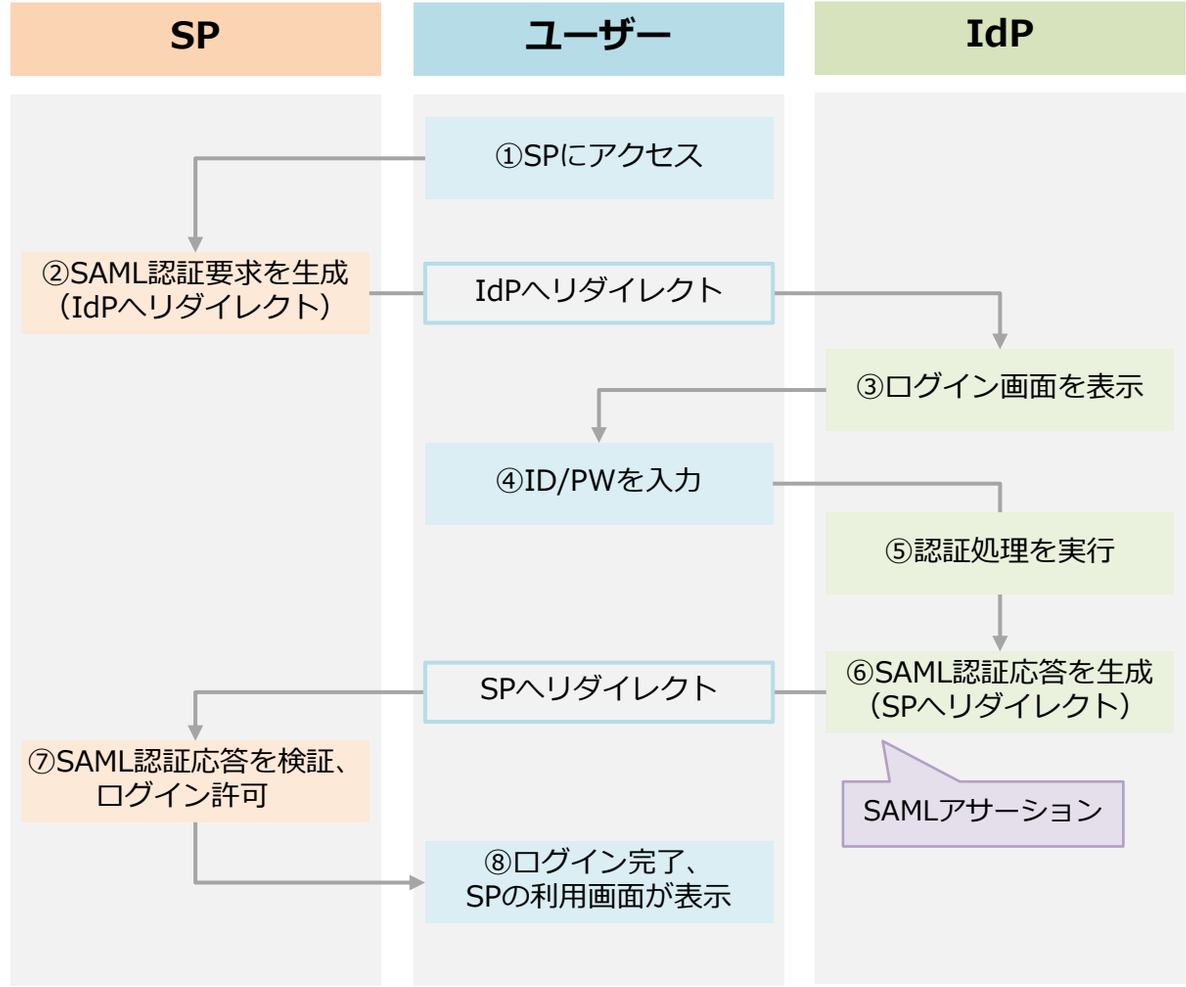
④ ユーザーはIdPのログイン画面で**認証情報を入力**する。

⑤ IdPは入力された情報をもとに**ユーザーの認証を行う**。

⑥ 認証が成功したら、IdPはSAMLアサーションを含む、**SAML認証応答を生成**する。アサーションにはユーザーの属性情報や認証状態、認可情報、デジタル署名などが含まれる。
IdPはSAML認証応答を、**ユーザーのブラウザ経由でSPにリダイレクトして送信**する。

⑦ SPは受け取った**SAML認証応答を検証**し、発行元（IdP）が信頼できるか、有効期限が切れていないか、ユーザー属性が正しいかなどを確認する。
検証により、アサーションの改ざんを防止し、SAML認証の信頼性を担保している。

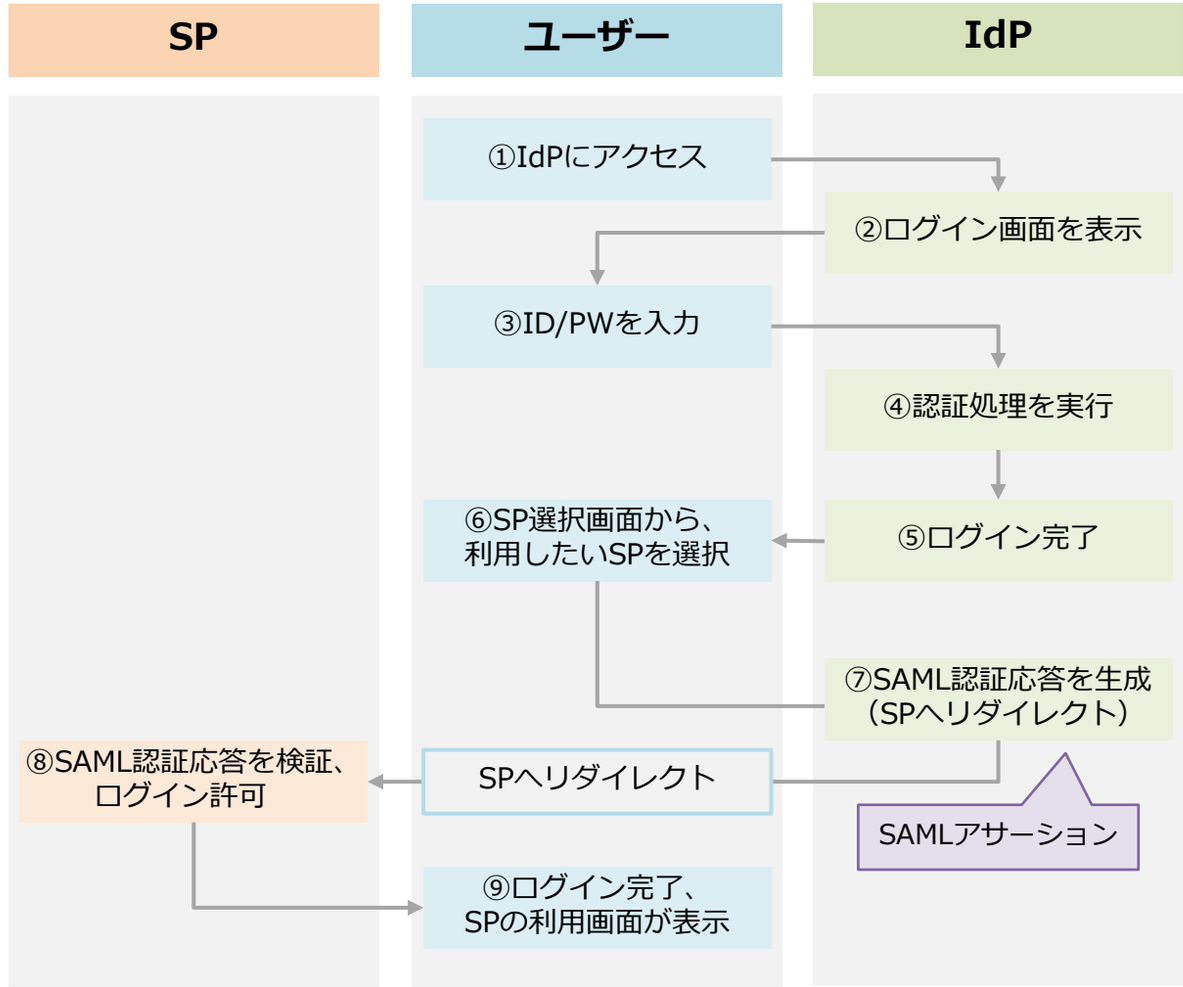
⑧ すべての検証が完了すると、SPはユーザーを認証済みとして扱い、**ユーザーはSPのサービス画面にアクセス**できるようになる。



# 3.2.5. SAML認証のフロー（IdP起点）

## SAML認証のフロー：IdP起点

① ユーザーがWebブラウザから**IdPのポータルサイトにアクセス**する。
② IdPはユーザーに**ログイン画面を表示**し、認証情報（IDとパスワードなど）の入力を求める。
③ ユーザーはIdPのログイン画面で**認証情報を入力**する。
④ IdPは入力された情報をもとに**ユーザーの認証を行う**。
⑤ 認証に成功すると、**IdPへのログインが完了**する。
⑥ ユーザーはIdPのポータル上で利用可能なSPの一覧から、**アクセスしたいSPを選択**する。
⑦ IdPはSAMLアサーションを含む**SAML認証応答を生成**する。
アサーションには、ユーザーの属性情報、認証状態、認可情報、デジタル署名などが含まれる。
IdPはSAML認証応答を、**ユーザーのブラウザ経由でSPにリダイレクトして送信**する。
⑧ SPは受け取った**SAML認証応答を検証**し、発行元（IdP）が信頼できるか、有効期限が切れていないか、ユーザー属性が正しいかなどを確認する。
検証により、アサーションの改ざんを防止し、SAML認証の信頼性を担保している。
⑨ すべての検証が完了すると、SPはユーザーを認証済みとして扱い、**ユーザーはSPのサービス画面にアクセス**できるようになる。



# 3.3. OIDC

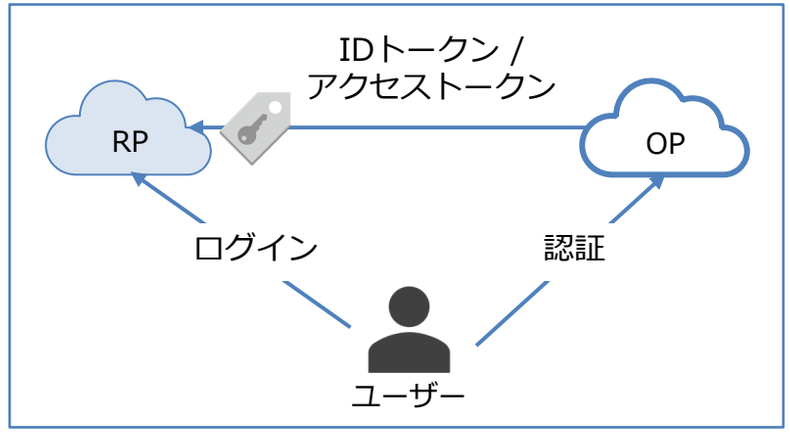
# 3.3.1. OIDCとは

## OIDC（OpenID Connect）とは

**OAuth 2.0をベースにした認証プロトコルで、ユーザーの同意のもと、認証情報や属性情報を他のサービスと安全に連携**できます。
**SSOを実現**し、Webサービスやアプリ間のスムーズなログインを可能にします。

**特徴**　Oauth.2.0については、後段のスライドで説明

- **OAuth 2.0を拡張した認証プロトコル**
  OIDCはOAuth 2.0をベースに**「認可」に加えて「認証」も扱えるように拡張されたプロトコル**です。
  アクセストークンによるAPIアクセスに加え、IDトークンでユーザーの認証情報も取得できます。
- **OP（OpenID Provider）とRP（Relying Party）間の認証連携**
  OIDCは、**認証を行うOP**（例：Entra ID）と、**サービスを提供するRP**（例：Webサービス）が連携して動作します。
  ユーザーがOPで認証されると、その認証結果（IDトークン）がRPに送信され、ユーザーはパスワードを再入力することなく
  サービスを利用できます（SSOを実現）。
- **JSON Web Token (JWT) の利用**
  認証結果や属性情報は、**IDトークンとしてJWT形式で発行**されます。
  JWTはデジタル署名により改ざんを防止でき、発行元の検証も可能です。
- **API連携に最適**
  JSONベースで軽量かつ構造がシンプルなため、**Web APIやモバイルアプリ、**
  **SPA**などのモダンな開発環境と親和性が高いです。



ユーザー体験

# 3.3.1. OIDCとは

## 利用シーン

**✓ モバイルアプリでのソーシャルログイン**

Microsoftアカウントで外部サービスにログイン

- 外部サービス（例：Uber、Spotify）がIdP（例：Microsoft）とOIDC連携することで、ユーザーは新たにアカウントを作らずにログイン可能。
- → パスワード管理の負担軽減とユーザー体験の向上に寄与。

**✓ 社内クラウドサービスのSSO**

Microsoft 365、Box、Slackなどのクラウドサービスと連携

- ユーザーは社内の認証システム（例：Entra ID、Okta）で一度ログインするだけで、すべての連携サービスに自動ログイン可能。
- → パスワード入力の手間を省き、ユーザー体験とセキュリティを同時に向上。

## メリット

- **導入が簡単**
  OIDCは標準プロトコルのため、短期間・少ない工数で実装可能。
- **業務効率化**
  SSOにより、一度のログインで複数サービスにアクセス。ID管理も一元化。
- **幅広い対応**
  Webアプリ、モバイルアプリ、APIなど多様なクライアントで利用可能。
- **高いセキュリティ**
- JWTによる改ざん防止、多要素認証や条件付きアクセスとの連携も容易。

## デメリットと対策

- **IDトークンの不正取得リスクと不正ログインの恐れ**
  → トークンの発行管理を厳重にし、通信の暗号化やアクセス制御を徹底。
- **OPが停止すると全サービスにログイン不可**
  → 認証基盤が単一障害点となるため、可用性の高いIdPの採用やフェールオーバー構成の検討が重要。

# 3.3.2. OIDCとOauth 2.0の関係性

## OAuth 2.0とは

外部サービスに安全にアクセス権を委ねるための「**認可**」のフレームワークです。
ユーザーのパスワードを渡さずに、**他のアプリやサービスに“アクセスの許可”を与える**仕組みです。

## OIDCとOauth 2.0の関係性

OIDC（OpenID Connect）は、**OAuth 2.0の認可フレームワークを拡張し、ユーザー認証の機能を追加した仕組み**です。
OAuth 2.0が**「何ができるか」（＝認可）をアクセストークン**で扱うのに対し、OIDCはそれに加えて**「誰であるか」（＝認証）をIDトークン**で扱います。


| 比較項目 | OAuth 2.0 | OpenID Connect (OIDC) |
|---|---|---|
| **目的** | 認可 | 認証＋認可 |
| **主な機能** | クライアントが保護リソース（APIなど）へアクセスするための権限付与。アクセストークンを発行。 | OAuth 2.0ベースで、ユーザーのID確認と属性情報を提供。OPがIDトークンを発行し、RPがユーザー身元を検証。 |
| **発行するトークン** | アクセストークン | IDトークンおよびアクセストークン |
| **利用例** | Outlookの予定表に外部アプリからアクセスし、予定表を読み書きする。 | シングルサインオン (SSO)、Web/モバイルアプリの認証 (Microsoft、Googleなど)。 |

# 3.3.3. OIDCの構成要素

OIDCは、ユーザー、認証を提供するサービス、利用するサービス、それらを連携するための情報から構成され、
これらの要素が次スライドで説明する認証フローの中で重要な役割を果たします。

| 要素 | 説明 |
|---|---|
| **OP（OpenID Provider）** | **ユーザーの認証情報や属性、権限を管理し、認証を行うシステム。**ユーザーのログイン情報はOPが保持し、RPには認証結果を連携する。<br>これにより、RPはパスワードを持たず、認証はすべてOP側で行われる。　例：Microsoft Entra ID、Google Workspace、Oktaなど。 |
| **RP (Relying Party)** | **ユーザーにサービスを提供する側のシステム（クライアントアプリケーション）**。<br>OPで認証されたユーザー情報を受け取り、サービスを提供する。OPから受け取るIDトークンやアクセストークンを活用し、<br>ユーザーの認証状態を確認したり、ユーザーのデータにアクセスすることが可能。　例：Webアプリケーション、モバイルアプリなど。 |
| **ID トークン** | OPが発行する**JWT形式の認証情報**。ユーザーの認証状態や属性情報（名前、メールアドレスなど）、デジタル署名などの情報を持つ。RPはこれを検証することで、ユーザーが正しく認証されたことを確認できる。<br>IDトークンの有効期限はOPで設定され、Entra IDでは通常60分が既定である。 |
| **アクセストークン** | OAuth 2.0の仕組みに基づき、**RPがユーザーに代わって保護されたリソース（APIなど）にアクセスするための許可証。**<br>特定のスコープ（アクセス範囲）と有効期限を持つ。有効期限はOPで設定され、Entra IDでは通常60～90分（平均75分）が既定である。<br>IDトークンとは異なり、主に**リソースへのアクセス権限**を示す。 |
| **クライアントシークレット** | RP自身をOPが認証するための「**秘密鍵**」のようなもの。クライアントシークレットは、特にサーバーサイドで動作するアプリケーション（Confidential Client）が、OPからトークンを取得する際に、そのRPが正当なRPであることを証明するために使用される。 |

**OPとRPの信頼関係**

ユーザー認証前に、クライアント登録という形でRPがOPに事前に
登録されている（信頼関係を作る）必要があります。
その際にクライアントIDやリダイレクトURLの情報を登録します。
これにより、認証処理は安全かつ正しい相手とだけ行われます。

**アクセストークンの補足**

ユーザーの代わりにサービス（APIなど）へアクセスするための「**通行証**」です。
ユーザーの認証後に発行され、パスワードを渡さずに安全に操作できます。
例：Microsoftアカウントでログインしたカレンダーアプリが、
アクセストークンを使ってOutlookカレンダーの予定を取得する

# 3.3.4. OIDCの認証フロー

最も一般的な**認可コードフロー**をベースに、OIDC認証フローについて解説します。

① ユーザーが**RPのWebサイトやアプリにアクセスし、認証を開始**する。
例）RPの画面で「Microsoftでログイン」のリンクをクリックする

② ID連携の要求を受けたRPは、**認証リクエストを生成する。**
**ユーザーのブラウザをOPへリダイレクト**させ、認証リクエストを送信する。

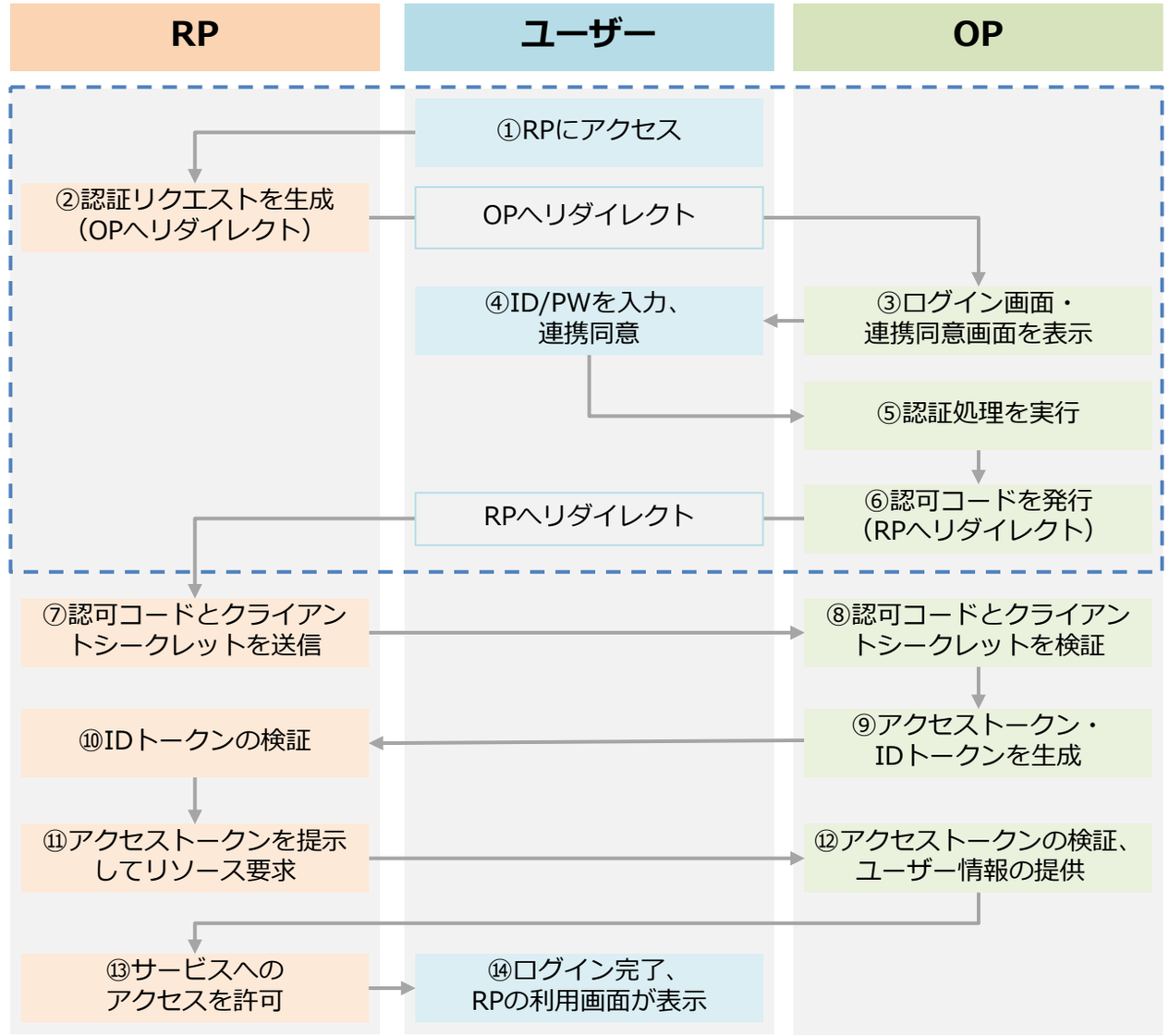
③ OPはユーザーに**ログイン画面を表示**し、認証情報（IDとパスワードなど）の入力を求める。
必要に応じて、RPがアクセスを求めているユーザー情報（例：氏名、生年月日、住所など）をユーザーに提示し、**その情報へのアクセスを許可するかどうかの同意を求める。**

④ ユーザーはログイン画面で**認証情報を入力**し、RPへの**情報連携に同意**する。

⑤ OPは入力された情報をもとに**ユーザーの認証を行う。**

⑥ 認証が成功すると、**OPはRPに「認可コード」を発行**する。
**ユーザーのブラウザをRPへリダイレクト**させ、**認可コードを送信**する。

認可コードとは、ユーザーがOPで認証されたことを一時的に示すコードで、最終的にIDトークン・アクセストークンと交換するために使われます。



# 3.3.4. OIDCの認証フロー

⑦ RPは受け取った**認可コード**と、自身の**クライアントシークレットをOPに送信**して、トークンをリクエストする。

⑧ OPは、受け取った**認可コードとクライアントシークレットを検証**する。
検証では認可コードの有効性や正しいリクエストか、などを確認する。

⑨ OPは、検証が完了したら「**IDトークン**」と「**アクセストークン**」を生成しRPに返す。

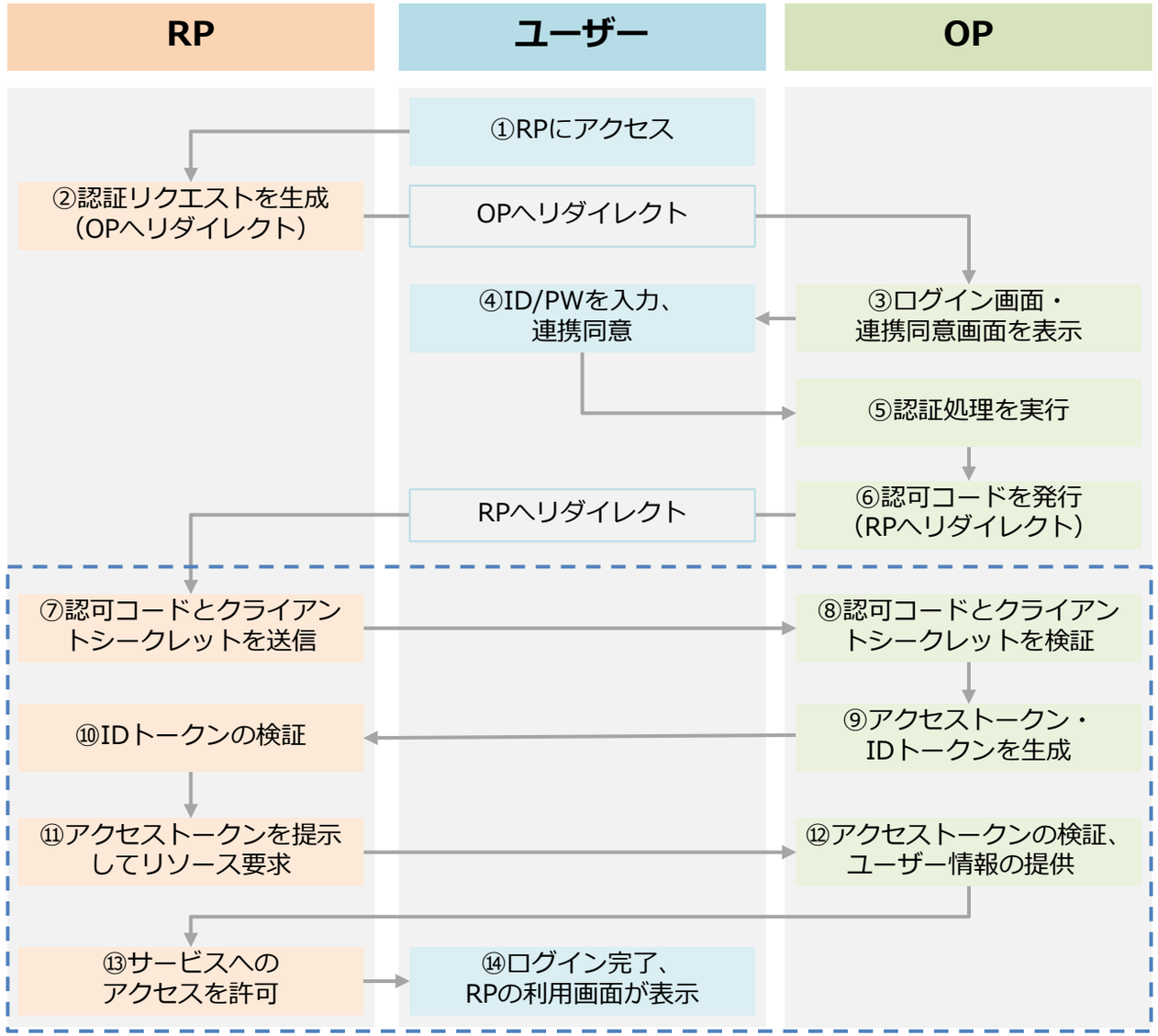
⑩ RPは受け取った**IDトークンのデジタル署名を検証**し、改ざんされていないこと、有効期限内であること、正規のOPから発行されたものであることを確認する。

⑪ RPは、必要に応じて**アクセストークンを提示**し、**OPに対して追加のユーザー情報取得やリソースへのアクセスを要求**。 （例：氏名、生年月日、住所など）

⑫ OPは、RPから受け取ったアクセストークンの有効性を検証し、問題なければ**アクセスが許可されているユーザー情報（属性など）をRPに提供**する。
フローの⑦～⑪はユーザーを介さず、RPとOP間で直接行われる。

⑬ PRは、情報取得が完了するとユーザーを認証済みとして扱い、**サービス画面へのアクセスを許可**する。
ユーザー情報の取得後、RPはアクセス制御をしたりプロフィールページの自動生成を行う。

⑭ ユーザーは**RPのサービスを利用できる**ようになる。



# 3.4. SAMLとOIDCの比較

SSOや外部サービスとの連携を実現するために、多くの企業やサービスで採用されているのが SAML や OIDC（OpenID Connect） です。
これらはどちらも「認証」「認可」を扱う仕組みですが、仕組みや用途、対応するシステムには明確な違いがあります。
本章では、それぞれの基本的な機能やフローを解説した上で、最後にSAMLとOIDCの違いを一覧で比較できる表を掲載しています。

| 比較項目 | SAML | OpenID Connect (OIDC) |
|---|---|---|
| **目的** | 認証と属性情報の提供 | OAuth 2.0を拡張して認証機能を提供 |
| **技術基盤** | XMLベースで、SOAPやHTTPなどを使用 | OAuth 2.0をベースとし、JSON (JWT) を使用 |
| **トークン形式** | SAMLアサーション（XML） | IDトークン（JWT）、アクセストークン（JWT等） |
| **提供情報** | ユーザーの認証情報（属性情報含む）と認可情報（ロールなど）をSAMLアサーションで提供。 | ユーザーの認証情報（IDトークン）と、リソースアクセス認可（アクセストークン）を提供。 |
| **認可の扱い** | 属性やロールをアサーションに含めて対応可能 | OAuth 2.0のアクセストークンで対応可能 |
| **主な構成要素** | IdP（Identity Provider）/ SP（Service Provider） | OP（OpenID Provider）/ RP（Relying Party） |
| **主な用途** | エンタープライズSSOやクラウドサービスとの認証連携 | WebサービスやモバイルアプリのSSO、APIの認証 |
| **導入の柔軟性** | 実装が複雑、モバイルに不向き | 軽量で柔軟、モバイルアプリやSPAと親和性が高い |

**ポイント**

**SAML**：企業内システムとの連携に強く、社内SSOや業務アプリの認証に広く使われています。
**OIDC**：Webやスマホアプリと相性がよく、クラウドサービスやAPI連携での認証に向いています。
**★クラウドサービスを連携をする際は、IdPとSPの双方がSAMLやOIDCの認証プロトコルに対応しているかを事前に確認することが重要です。**