モバイル、オープンAPIを活用するための認証基盤の実装例

株式会社オージス総研　テミストラクトソリューション部

氏縄　武尊

© 2017 OGIS-RI Co., Ltd.

自己紹介

氏縄 武尊 (Ujinawa Takeru)

Work

株式会社オージス総研
テミストラクトソリューション部　4年目
認証・認可・ID管理・PKI
OpenID Foundation Japan EIWGメンバー

Favorite

Spec: OpenID Connect, OAuth2.0, SCIM
AWS: IAM, CloudTrail, Lambda

統合認証ソリューションを長年やってきました

ThemiStruct-WAM　シングルサインオン 認証基盤ソリューション

ThemiStruct-IDM　ID管理ソリューション

ThemiStruct-CM　電子証明書発行・管理 ソリューション

ワンタイムパスワードソリューション
ThemiStruct-OTP

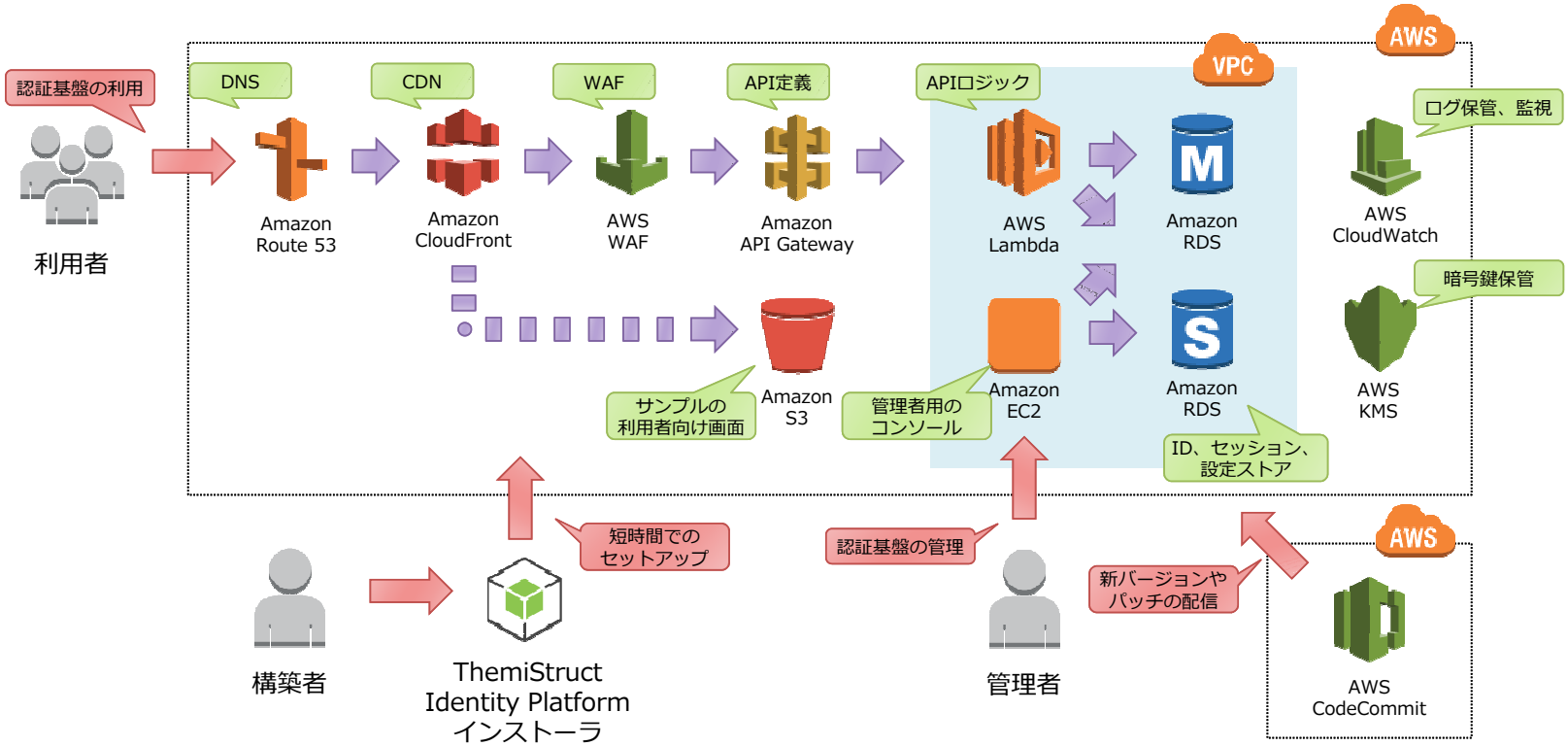
システム監視ソリューション
ThemiStruct-MONITOR

ThemiStruct（テミストラクト）
IdentityPlatform AWS対応版

クラウド、IoT時代の “All in one” 統合認証パッケージ

ThemiStruct Identity Platformの構成

ThemiStruct Identity Platform はAmazon Web Servicesのマネージドサービス上で稼働するため、
一定のアベイラビリティ、スケーラビリティを実現することができます。
また、以下の構成のセットアップを約2時間で完了できるインストーラを提供しています。

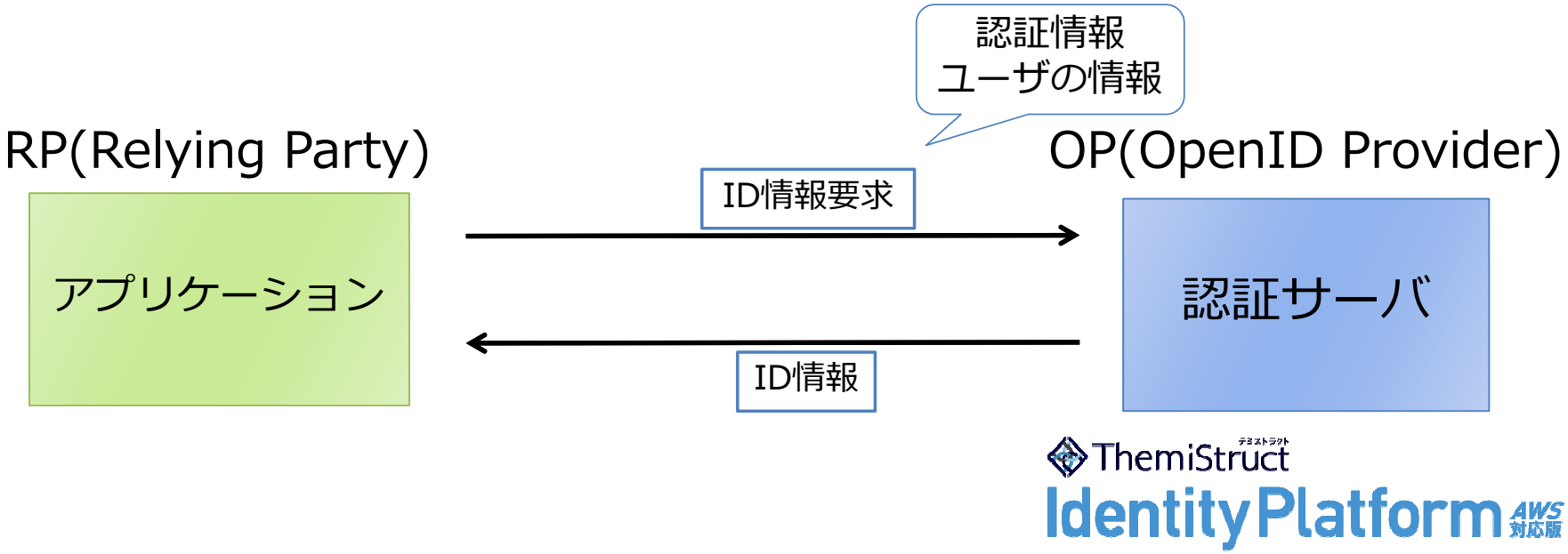


アイデンティティ連携を実現する標準技術たち

これらの技術は標準化が進み、製品・サービスへの実装も浸透している。

各種仕様の用語説明

OpenID Connect



認証情報
ユーザの情報

SAML SP(Service Provider)

SAML IdP(Identity Provider)

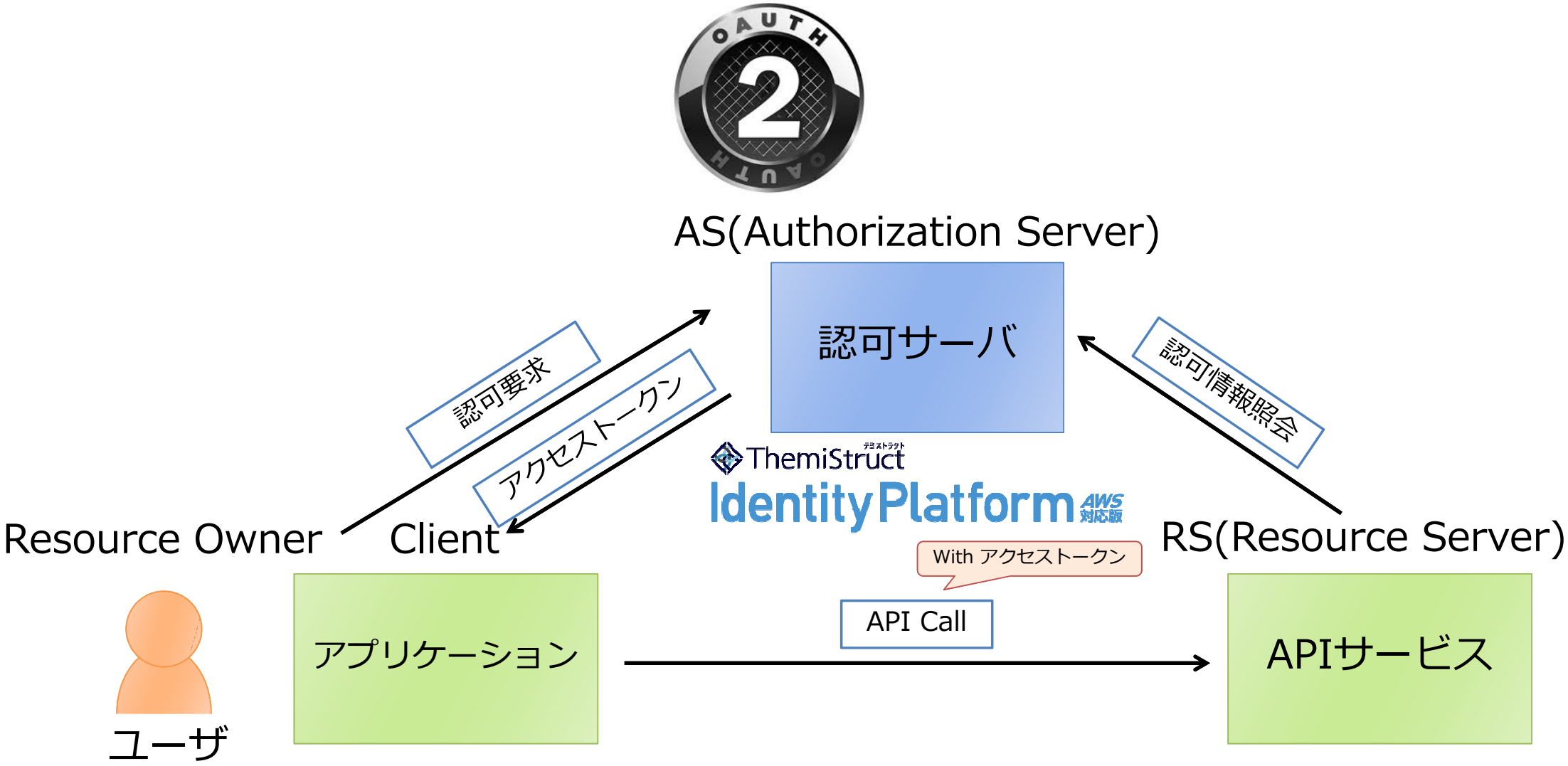
ID情報要求

アプリケーション

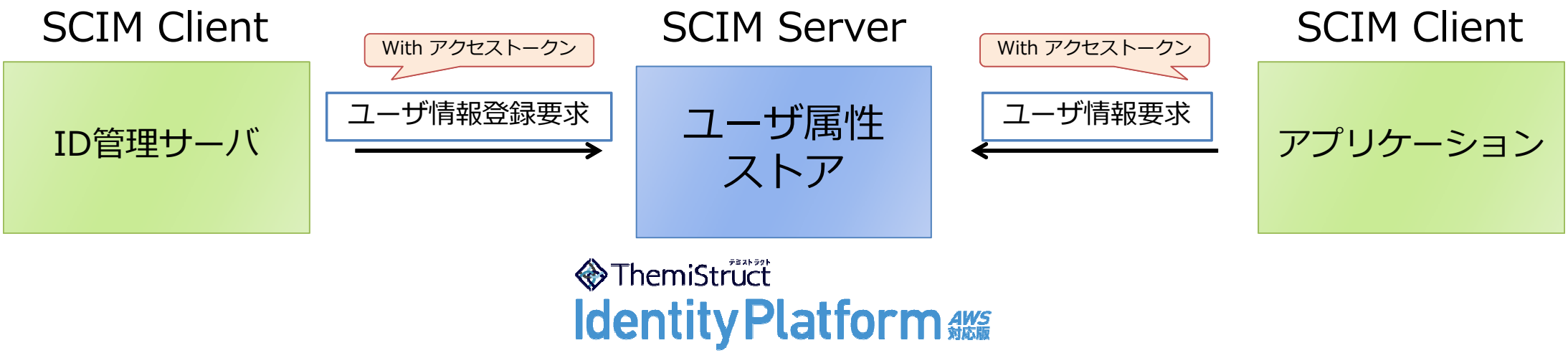
認証サーバ

ID情報

ThemiStruct
IdentityPlatform AWS対応版


AS(Authorization Server)
認可サーバ
認可要求
アクセストークン
認可情報照会
ThemiStruct
IdentityPlatform AWS対応版
Resource Owner
Client
RS(Resource Server)
With アクセストークン
API Call
アプリケーション
APIサービス
ユーザ

SCIM



ThemiStruct（テミストラクト）
IdentityPlatform AWS対応版

設定・実装例

モバイル向けアプリの導入・提供の形態

	従業員向けサービス	顧客向けサービス
外部サービスの利用 SaaSが提供するモバイル向けアプリを利用	○	―
モバイルウェブの開発と提供	○	○
ネイティブアプリの開発・提供	○	◎
サードパーティとの連携 APIを提供し、サードパーティがアプリを開発・提供	―	○

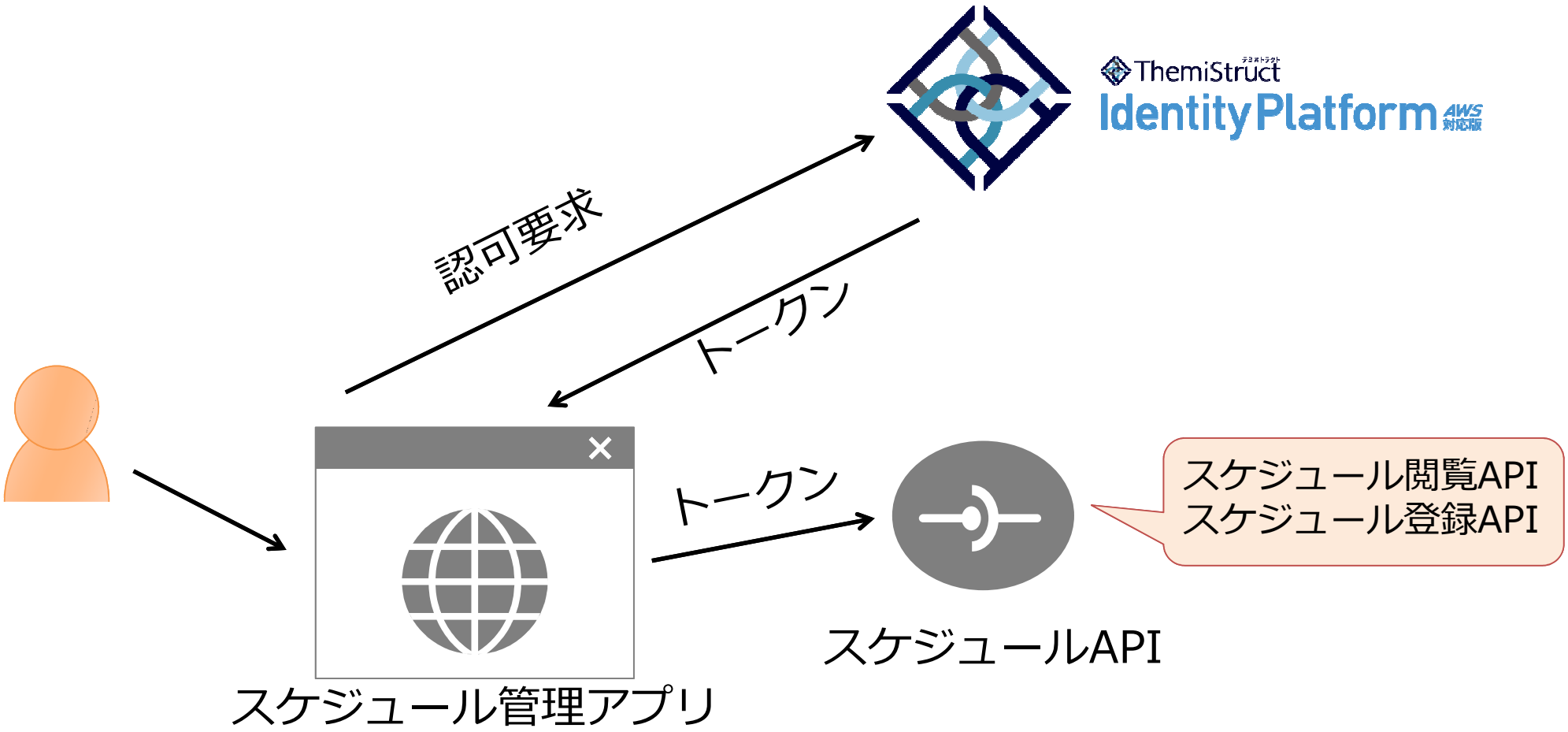
認証・アクセス認可の実装の考え方

	従業員向けサービス	顧客向けサービス
外部サービスの利用 SaaSが提供するモバイル向けアプリを利用	OpenID Connect や SAML を使ったSaaSへのSSO	—
モバイルウェブの開発と提供	OAuth 2.0 Implicit Grant を用いたバックエンドAPIへのアクセス認可	
ネイティブアプリの開発・提供	OAuth 2.0 for Native Apps のプラクティスを用いたバックエンドAPIへのアクセス認可	
サードパーティとの連携 APIを提供し、サードパーティがアプリを開発・提供	—	OAuth 2.0 を用いたAPI連携時認証･アクセス認可 Implicit, Authz Code, Authz Code+PKCE （OIDF FAPI WGの動向の考慮も必要）

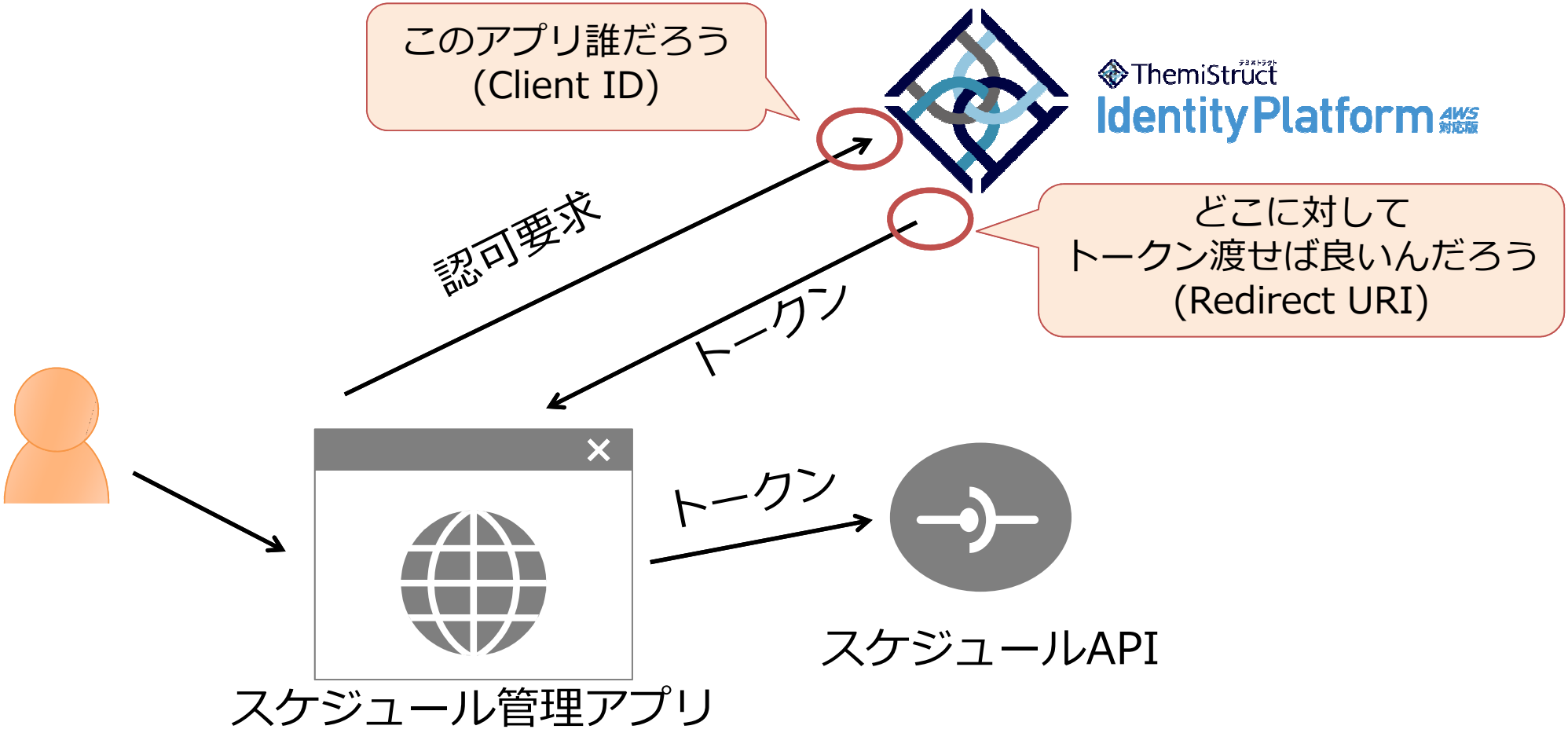
認証・アクセス認可の実装の考え方

	従業員向けサービス	顧客向けサービス
外部サービスの利用 SaaSが提供するモバイル向けアプリを利用	OpenID Connect や SAML を使ったSaaSへのSSO	—
モバイルウェブの開発と提供	OAuth 2.0 Implicit Grant を用いたバックエンドAPIへのアクセス認可	
ネイティブアプリの開発・提供	OAuth 2.0 for Native Apps のプラクティスを用いたバックエンドAPIへのアクセス認可	
サードパーティとの連携 APIを提供し、サードパーティがアプリを開発・提供	—	OAuth 2.0 を用いたAPI連携時認証･アクセス認可 Implicit, Authz Code, Authz Code+PKCE （OIDF FAPI WGの動向の考慮も必要）

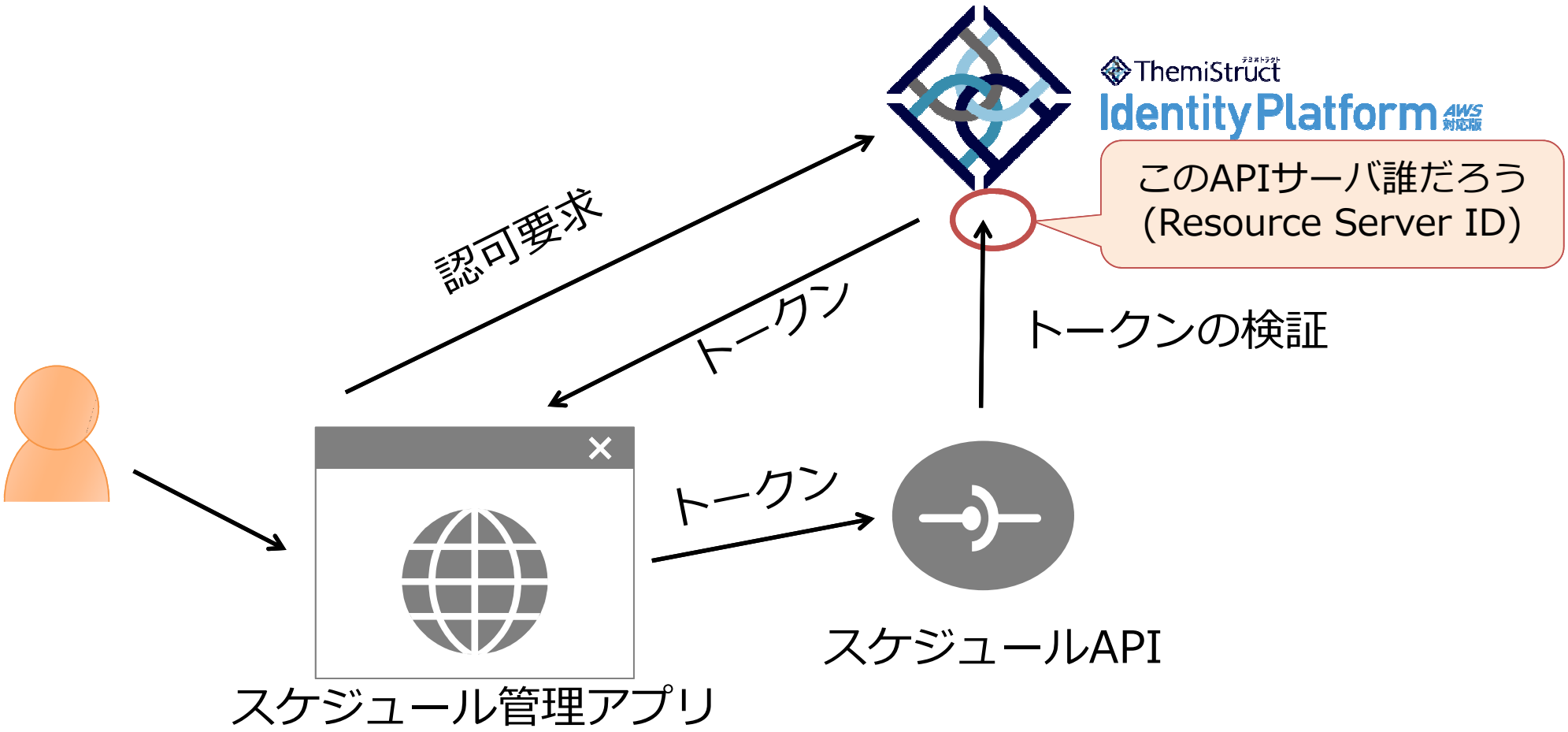
モバイルウェブアプリケーションの例



モバイルウェブアプリケーションの例



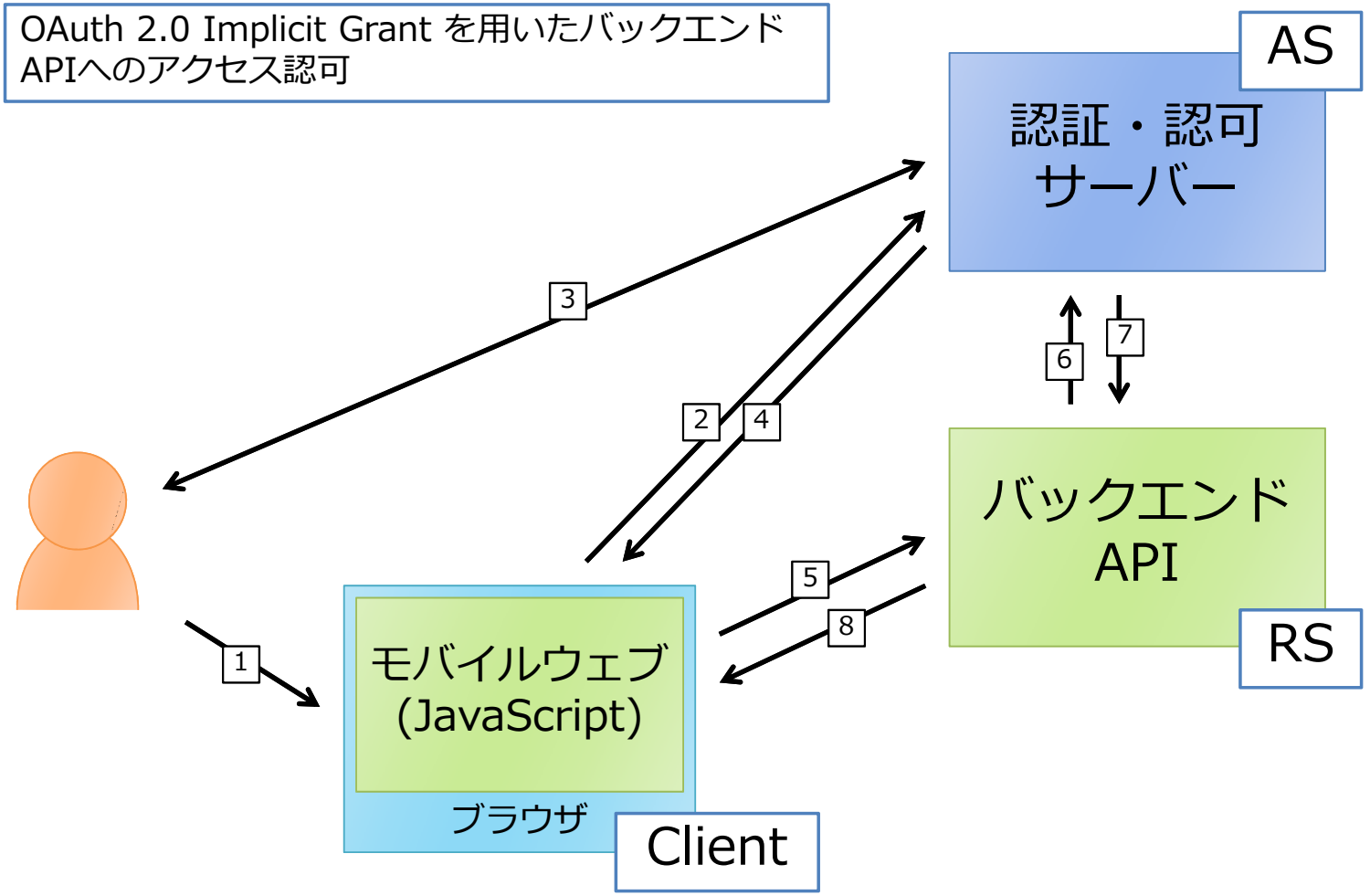
モバイルウェブアプリケーションの例



モバイルウェブでのバックエンドAPIアクセス認可

	従業員向けサービス	顧客向けサービス
外部サービスの利用 SaaSが提供するモバイル向けアプリを利用	○	—
モバイルウェブの開発と提供	○	○
ネイティブアプリの開発・提供	○	◎
サードパーティとの連携 APIを提供し、サードパーティがアプリを開発・提供	—	○

OAuth 2.0 Implicit Grant を用いたバックエンドAPIへのアクセス認可



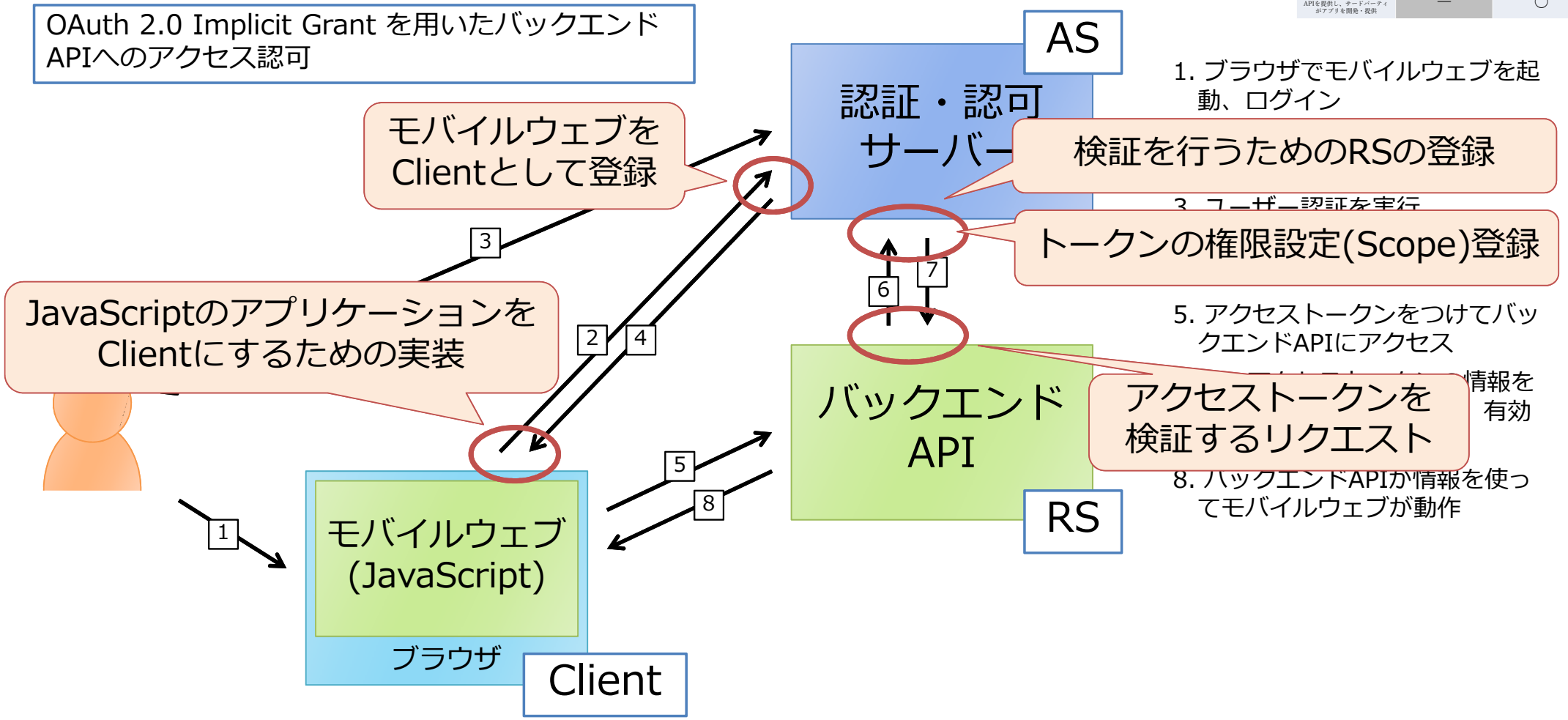
1. ブラウザでモバイルウェブを起動、ログイン
2. リダイレクトによりOAuth認可を要求
3. ユーザー認証を実行
4. バックエンドAPIアクセス用のアクセストークンを返却
5. アクセストークンをつけてバックエンドAPIにアクセス
6. ~7. アクセストークンの情報を確認（ユーザー、Scope、有効期限、など）
8. バックエンドAPIが情報を使ってモバイルウェブが動作

© 2017 OGIS-RI Co., Ltd.

モバイルウェブでのバックエンドAPIアクセス認可

	従業員向けサービス	顧客向けサービス
外部サービスの利用 SaaSが提供するモバイル向けアプリを利用	○	—
モバイルウェブの開発と提供	○	○
ネイティブアプリの開発・提供	○	◎
サードパーティとの連携 APIを提供し、サードパーティがアプリを開発・提供	—	○

OAuth 2.0 Implicit Grant を用いたバックエンドAPIへのアクセス認可



1. ブラウザでモバイルウェブを起動、ログイン

5. アクセストークンをつけてバックエンドAPIにアクセス

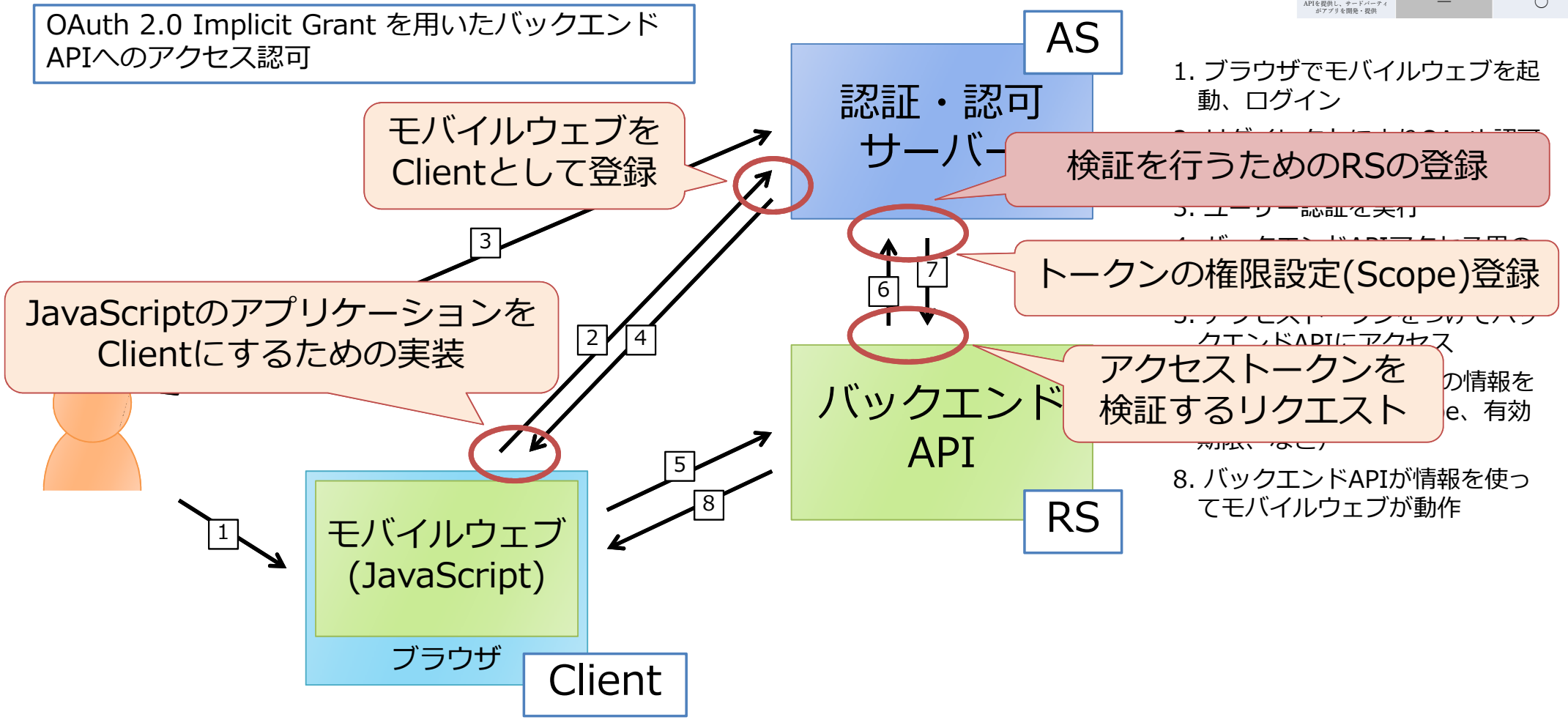
情報を有効

8. バックエンドAPIが情報を使ってモバイルウェブが動作

モバイルウェブでのバックエンドAPIアクセス認可

	従業員向けサービス	顧客向けサービス
外部サービスの利用 SaaSが提供するモバイル向けアプリを利用	○	—
モバイルウェブの開発と提供	○	○
ネイティブアプリの開発・提供	○	◎
サードパーティとの連携 APIを提供し、サードパーティがアプリを開発・提供	—	○

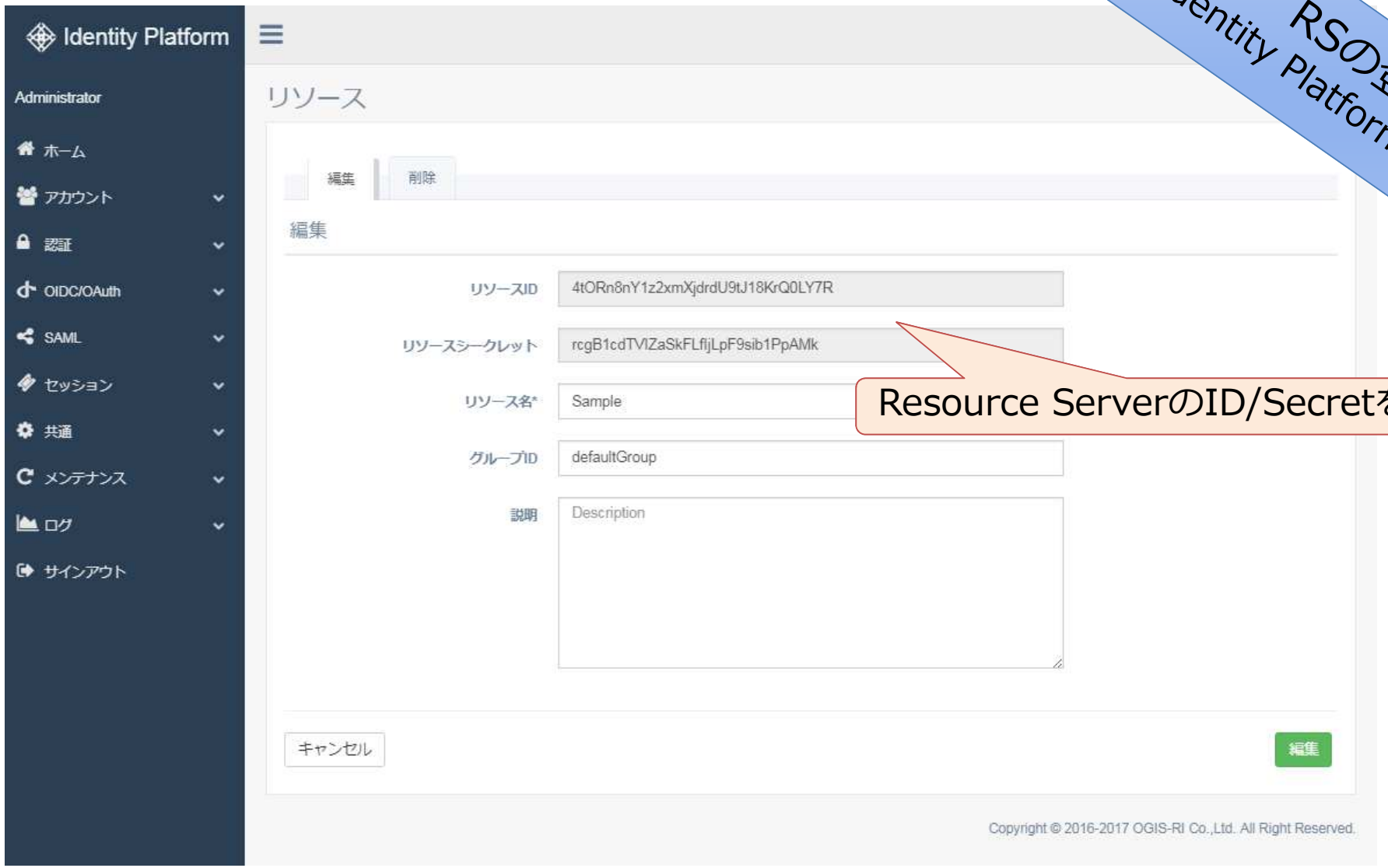
OAuth 2.0 Implicit Grant を用いたバックエンドAPIへのアクセス認可



1. ブラウザでモバイルウェブを起動、ログイン

8. バックエンドAPIが情報を使ってモバイルウェブが動作

RSの登録
Identity Platform管理画面

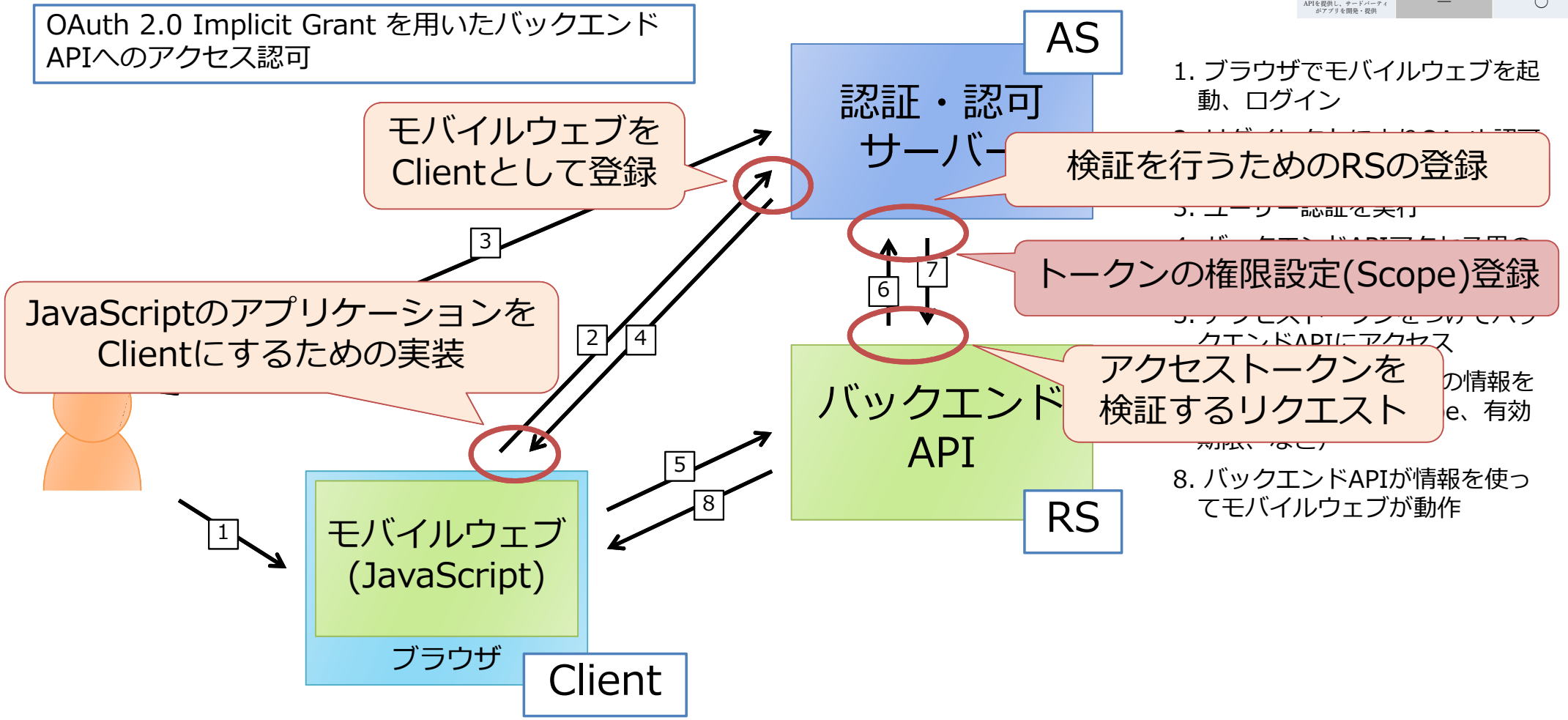


Resource ServerのID/Secretを確認

モバイルウェブでのバックエンドAPIアクセス認可

	従業員向けサービス	顧客向けサービス
外部サービスの利用 SaaSが提供するモバイル向けアプリを利用	○	—
モバイルウェブの開発と提供	○	○
ネイティブアプリの開発・提供	○	◎
サードパーティとの連携 APIを提供し、サードパーティがアプリを開発・提供	—	○

OAuth 2.0 Implicit Grant を用いたバックエンドAPIへのアクセス認可



1. ブラウザでモバイルウェブを起動、ログイン

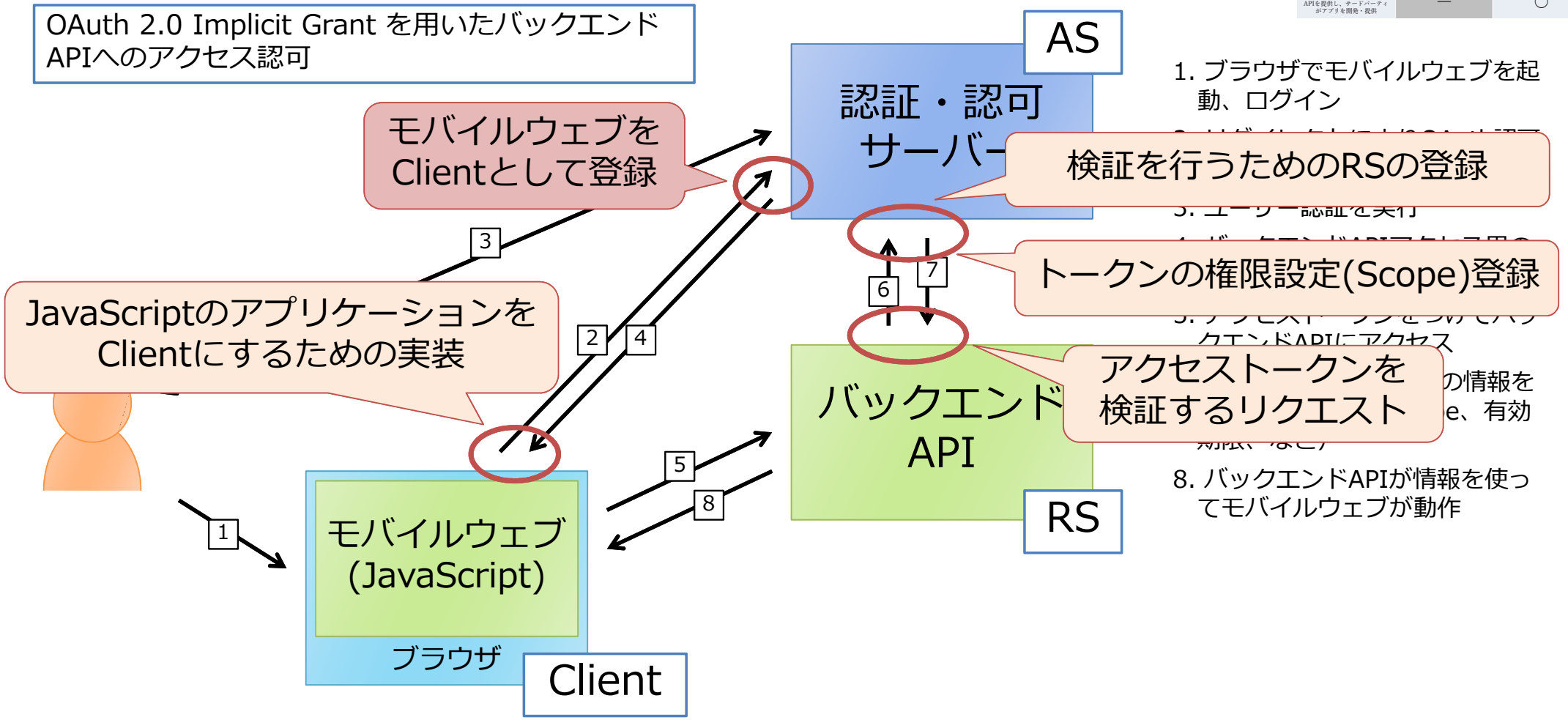
8. バックエンドAPIが情報を使ってモバイルウェブが動作

Scopeの登録
Identity Platform管理画面

モバイルウェブでのバックエンドAPIアクセス認可

	従業員向けサービス	顧客向けサービス
外部サービスの利用 SaaSが提供するモバイル向けアプリを利用	○	—
モバイルウェブの開発と提供	○	○
ネイティブアプリの開発・提供	○	◎
サードパーティとの連携 APIを提供し、サードパーティがアプリを開発・提供	—	○

OAuth 2.0 Implicit Grant を用いたバックエンドAPIへのアクセス認可



1. ブラウザでモバイルウェブを起動、ログイン
3. ユーザー認証を実行
5. アクセストークンを使ってバックエンドAPIにアクセス
の情報を
e、有効
8. バックエンドAPIが情報を使ってモバイルウェブが動作

Clientの登録
Identity Platform管理画面

リダイレクトURLを指定
（トークンの返し先）

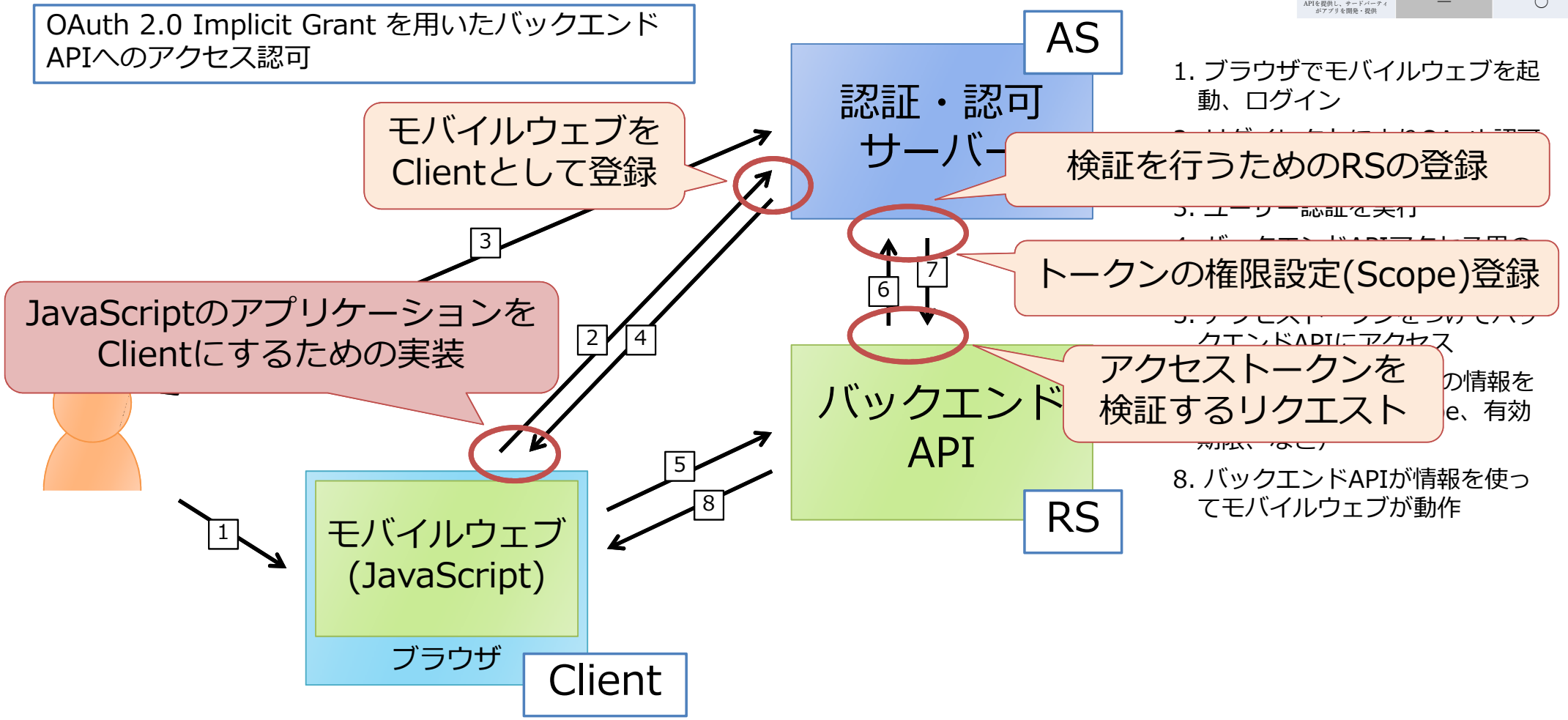
要求したいScopeも
指定する

登録が完了すると
ClientIDが発行される

モバイルウェブでのバックエンドAPIアクセス認可

	従業員向けサービス	顧客向けサービス
外部サービスの利用 SaaSが提供するモバイル向けアプリを利用	○	—
モバイルウェブの開発と提供	○	○
ネイティブアプリの開発・提供	○	◎
サードパーティとの連携 APIを提供し、サードパーティがアプリを開発・提供	—	○

OAuth 2.0 Implicit Grant を用いたバックエンドAPIへのアクセス認可



1. ブラウザでモバイルウェブを起動、ログイン
3. ユーザー認証を実行
5. クライアントトークンを使ってバックエンドAPIにアクセス
8. バックエンドAPIが情報を使ってモバイルウェブが動作

oidc-client-js
認可要求の設定

単純にアクセストークンを取得するだけなら
ライブラリ読み込み+100行弱のJavaScriptコードで実現可能

一部抜粋

```
<script src="oidc-client.js"></script>
```

```
<script>
var settings = {
    authority: 'https://example.com/oauth/v2/authz',
    client_id: 'SampleClientId',
    redirect_uri: 'https://rp.example.com/',
    response_type: 'token',
    scope: 'read write'
};

var manager = new Oidc.UserManager(settings);
```

パラメータの設定

oidc-client-js
認可要求・検証処理

単純にアクセストークンを取得するだけなら
ライブラリ読み込み+100行弱のJavaScriptコードで実現可能

一部抜粋

```
function authorizationRequest() {
  manager.signinRedirect() ;
}
```

ボタン選択時に呼び出されるように設定

```
if (window.location.hash.indexOf('#state=') == 0
    || window.location.hash.indexOf('&state=') > 0) {
  manager.signinRedirectCallback().then(function (u) {
    // ログイン成功時処理
  }).catch(function (e) {
    // ログイン失敗時処理
  }) ;
}
```

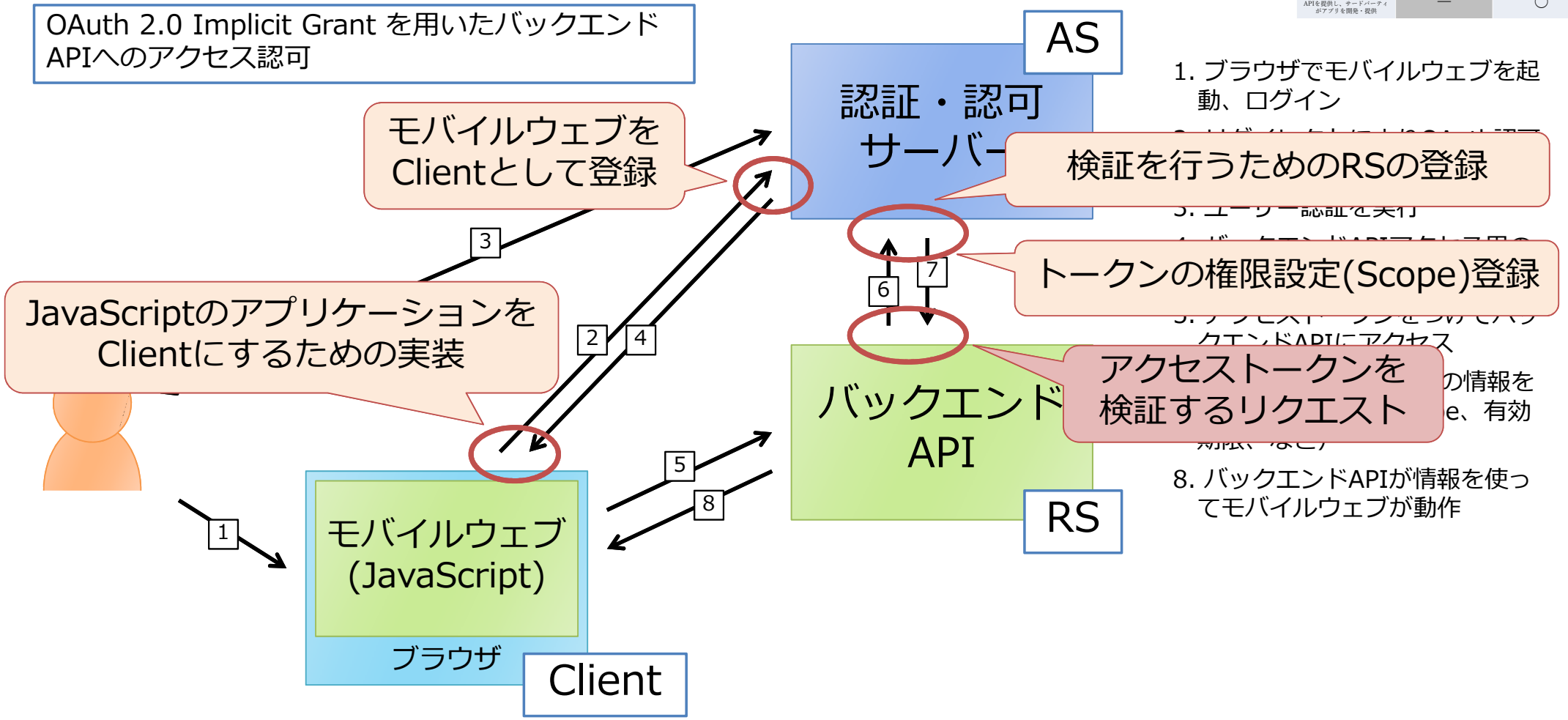
認可レスポンス返却時の処理

トークンの検証処理等は
ライブラリ内で行われる

モバイルウェブでのバックエンドAPIアクセス認可

	従業員向けサービス	顧客向けサービス
外部サービスの利用 SaaSが提供するモバイル向けアプリを利用	○	—
モバイルウェブの開発と提供	○	○
ネイティブアプリの開発・提供	○	◎
サードパーティとの連携 APIを提供し、サードパーティがアプリを開発・提供	—	○

OAuth 2.0 Implicit Grant を用いたバックエンドAPIへのアクセス認可



1. ブラウザでモバイルウェブを起動、ログイン
3. ユーザー認証を実行
5. クライアントトークンをつかってバックエンドAPIにアクセス
8. バックエンドAPIが情報を使ってモバイルウェブが動作

トークンの検証

TokenIntrospection
リクエスト

```
curl "https://ip.example.com/oauth/v2/token/introspect" ¥
-X POST ¥
-H "Content-Type:application/x-www-form-urlencoded" ¥
-H "Authorization: Basic U2FtcGxlUlNJRDpTYW1wbGVSU1NlY3JldA==" ¥
-d "token=SampleAccessToken" -i'
```

ResourceID, SecretのBasic認証ヘッダ

検証したいアクセストークン

Token Introspection
レスポンス

```
{
  "active": true,
  "client_id": "Gvg5PtkkPIHpPRiAXCZluxcHYaSrGO8q",
  "token_type": "Bearer",
  "scope": [
    "read",
    "write"
  ],
  "sub": "Azv9E0aLI1s6JVbAuzCAzPR9k2b0KiZQyOxJiuPLiy52",
  "iss": "https://ip.example.com/oauth/v2",
  "exp": 1507803616,
  "iat": 1507800016,
  "extension_field": ""
}
```

トークンの検証

TokenIntrospection
リクエスト

```
curl "https://ip.example.com/oauth/v2/token/introspect" ¥
-X POST ¥
-H "Content-Type:application/x-www-form-urlencoded" ¥
-H "Authorization: Basic U2FtcGxlUlNJRDpTYW1wbGVSU1NlY3JldA==" ¥
-d "token=SampleAccessToken" -i'
```

Token Introspection
レスポンス

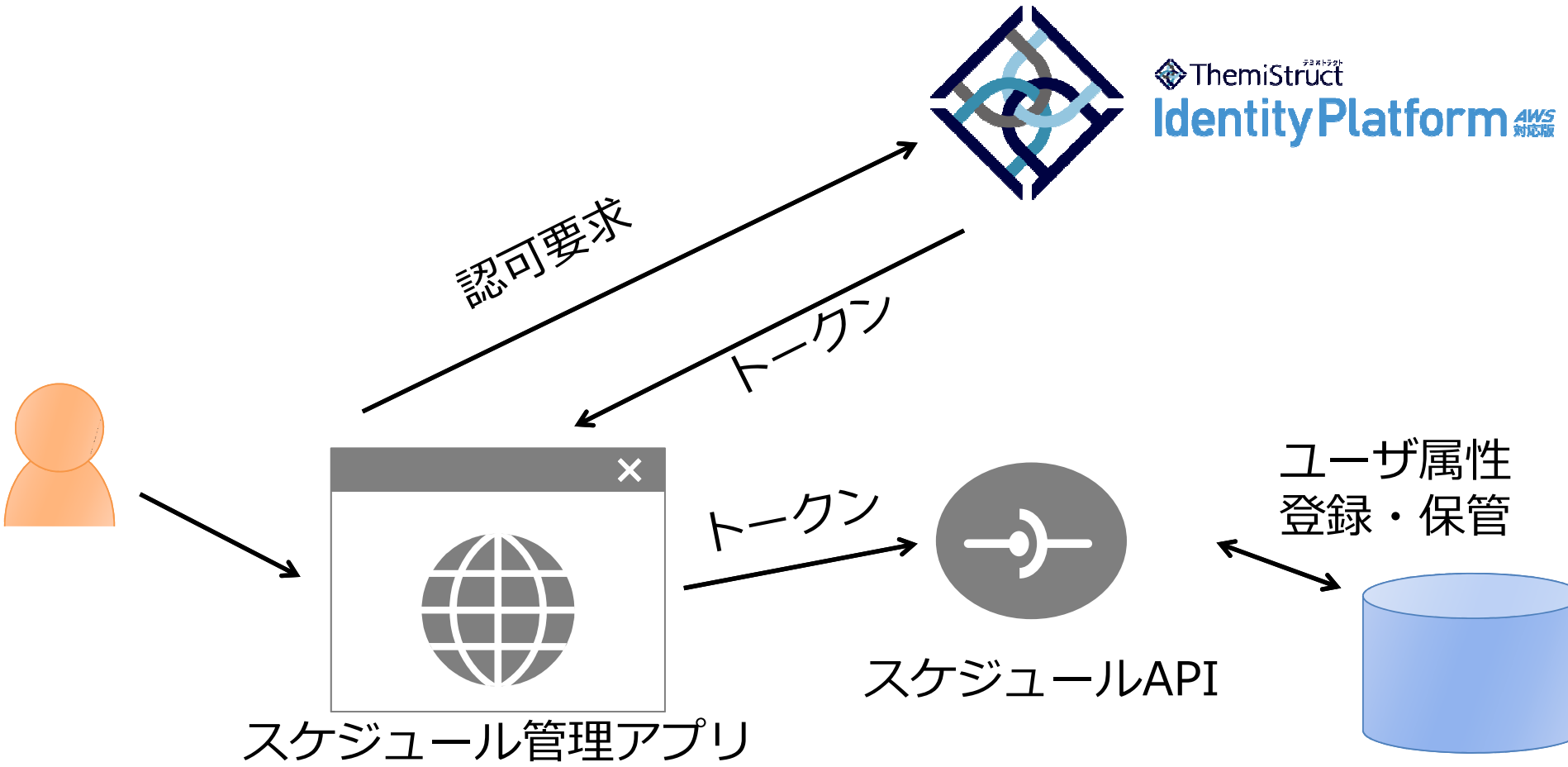
```
{
  "active": true,
  "client_id": "Gvg5PtkkPIHpPRiAXCZluxcHYaSrGO8q",
  "token_type": "Bearer",
  "scope": [
    "read",
    "write"
  ],
  "sub": "Azv9E0aLI1s6JVbAuzCAzPR9k2b0KiZQyOxJiuPLiy52",
  "iss": "https://ip.example.com/oauth/v2",
  "exp": 1507803616,
  "iat": 1507800016,
  "extension_field": ""
}
```

許可された権限

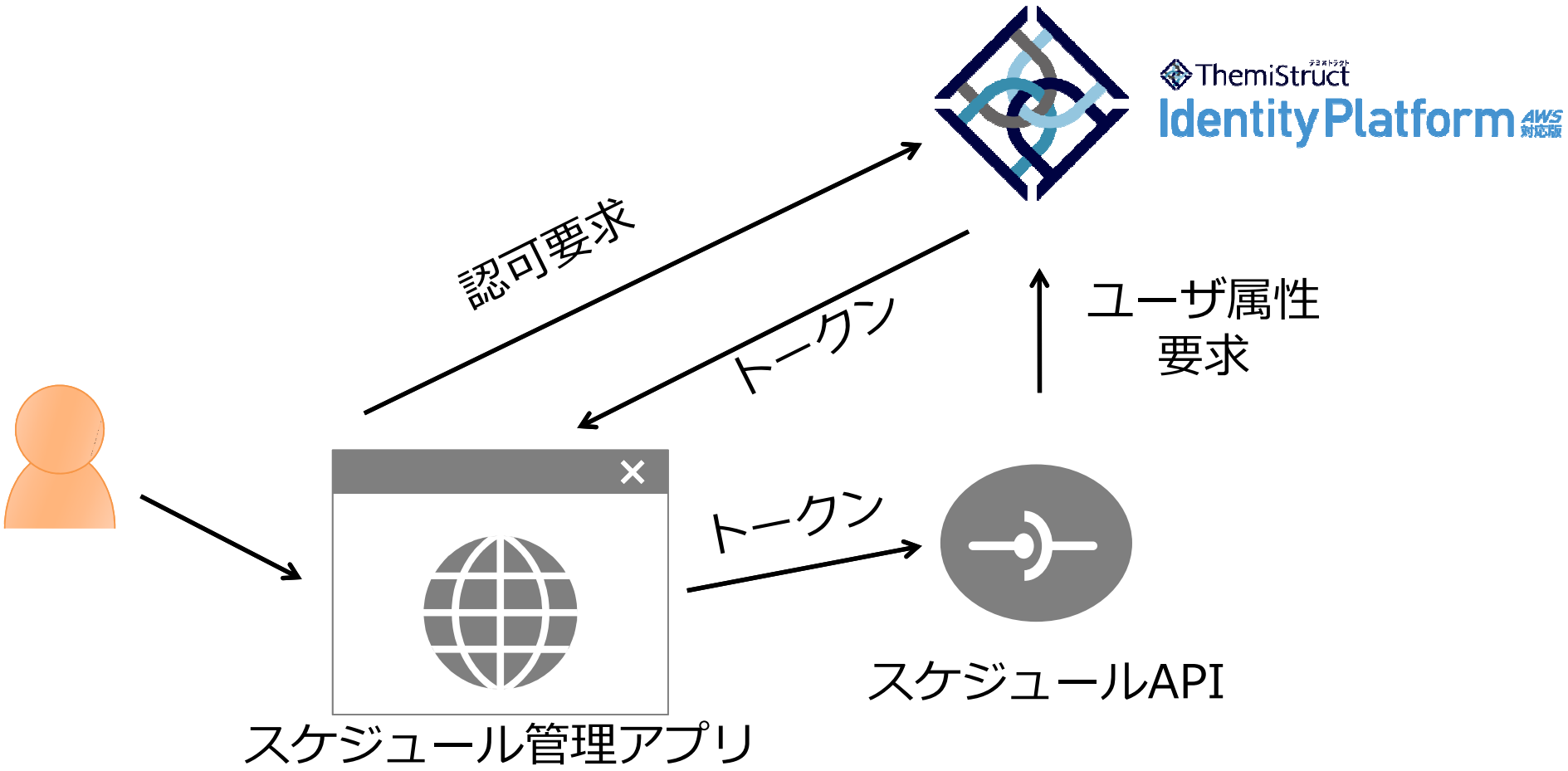
ユーザの識別子

期限

モバイルウェブアプリケーションの例



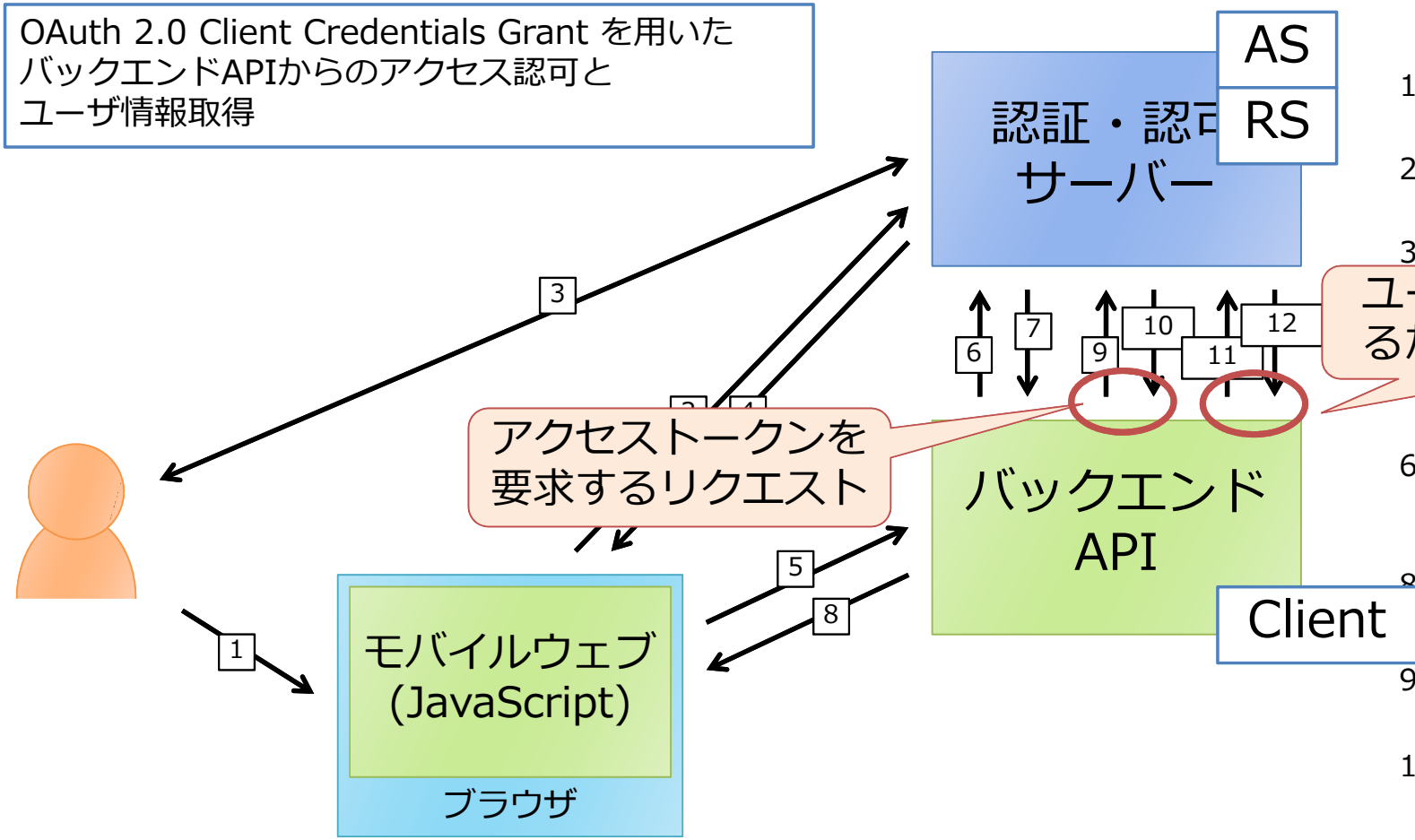
モバイルウェブアプリケーションの例



モバイルウェブでのバックエンドAPIアクセス認可

	従業員向けサービス	顧客向けサービス
外部サービスの利用 SaaSが提供するモバイル向けアプリを利用	○	—
モバイルウェブの開発と提供	○	○
ネイティブアプリの開発・提供	○	◎
サードパーティとの連携 APIを提供し、サードパーティがアプリを開発・提供	—	○

OAuth 2.0 Client Credentials Grant を用いた
バックエンドAPIからのアクセス認可と
ユーザ情報取得



1. ブラウザでモバイルウェブを起動、ログイン
2. リダイレクトによりOAuth認可を要求
3. ユーザー認証を実行
4. [illegible]用のアクセストークンを[illegible]てバックエンドAPIにアクセス
6.~7. アクセストークンの情報を確認（ユーザー、Scope、有効期限、など）
8. バックエンドAPIが情報を使ってモバイルウェブが動作
9.~10. バックエンドAPIがOAuth認可を要求
11.~12. バックエンドAPIがユーザ属性を要求

アクセストークン要求

トークン
リクエスト

```
curl "https://ip.example.com/oauth/v2/token" ¥
-X POST ¥
-H "Content-Type:application/x-www-form-urlencoded" ¥
-H "Authorization: Basic U2FtcGxlQ2xpZW50SUQ6U2FtcGxlQ2xpZW50U2VjcmV0" ¥
-d "grant_type=client_credentials&scope=tsipacouunt:get"
```

クライアント用のトークンをリクエスト

ユーザ取得のための権限を要求

トークン
レスポンス

```
{
  "access_token":"SampleAccessToken",
  "token_type":"bearer",
  "expires_in":"600"
}
```

ユーザ情報要求

ユーザ
リクエスト

```
curl "https://ip.example.com/account/v2/Users/SampleUserId" ¥
-X GET ¥
-H "Content-Type:application/scim+json" ¥
-H "Authorization: Bearer SampleAccessToken"
```

取得したアクセストークンでユーザ情報を要求

ユーザ
レスポンス

```
{
  "schemas": [
    "urn:ietf:params:scim:schemas:core:2.0:User"
  ],
  "id": "SampleUserId",
  "userName": "SampleUser",
  "nickName": "sample",
  "active": true,
      ・
      ・
      ・
```

認証・アクセス認可の実装の考え方

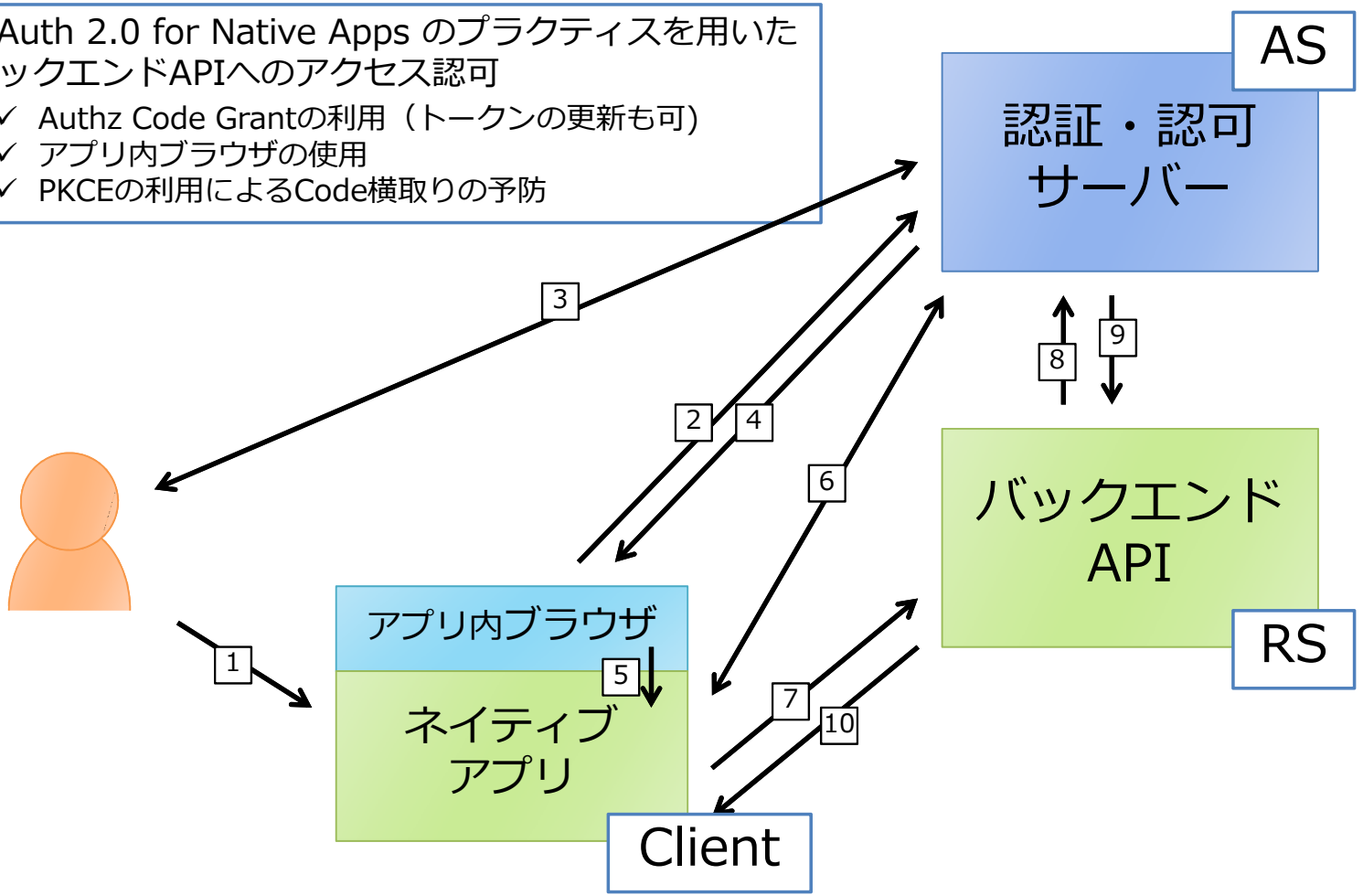
	従業員向けサービス	顧客向けサービス
外部サービスの利用 SaaSが提供するモバイル向けアプリを利用	OpenID Connect や SAML を使ったSaaSへのSSO	—
モバイルウェブの開発と提供	OAuth 2.0 Implicit Grant を用いたバックエンドAPIへのアクセス認可	
ネイティブアプリの開発・提供	OAuth 2.0 for Native Apps のプラクティスを用いたバックエンドAPIへのアクセス認可	
サードパーティとの連携 APIを提供し、サードパーティがアプリを開発・提供	—	OAuth 2.0 を用いたAPI連携時認証･アクセス認可 Implicit, Authz Code, Authz Code+PKCE （OIDF FAPI WGの動向の考慮も必要）

ネイティブアプリでのバックエンドAPIアクセス認可

	従業員向けサービス	顧客向けサービス
外部サービスの利用 SaaSが提供するモバイル向けアプリを利用	○	—
モバイルウェブの開発と提供	○	○
ネイティブアプリの開発・提供	○	◎
サードパーティとの連携 APIを提供し、サードパーティがアプリを開発・提供	—	○

OAuth 2.0 for Native Apps のプラクティスを用いたバックエンドAPIへのアクセス認可

- ✓ Authz Code Grantの利用（トークンの更新も可）
- ✓ アプリ内ブラウザの使用
- ✓ PKCEの利用によるCode横取りの予防



1. ネイティブアプリを起動、ログイン
2. アプリ内ブラウザを使いOAuth認可を要求（PKCEを併用）
3. ユーザー認証を実行
4. アクセストークンを取得するためのCodeを返却
5. アプリ内ブラウザからネイティブアプリにCodeを返却
6. Codeをアクセストークンに交換（PKCEを併用）
7. アクセストークンをつけてバックエンドAPIにアクセス
8. ~9. アクセストークンの情報を確認（ユーザー、Scope、有効期限、など）
10. バックエンドAPIが情報を使ってネイティブアプリが動作

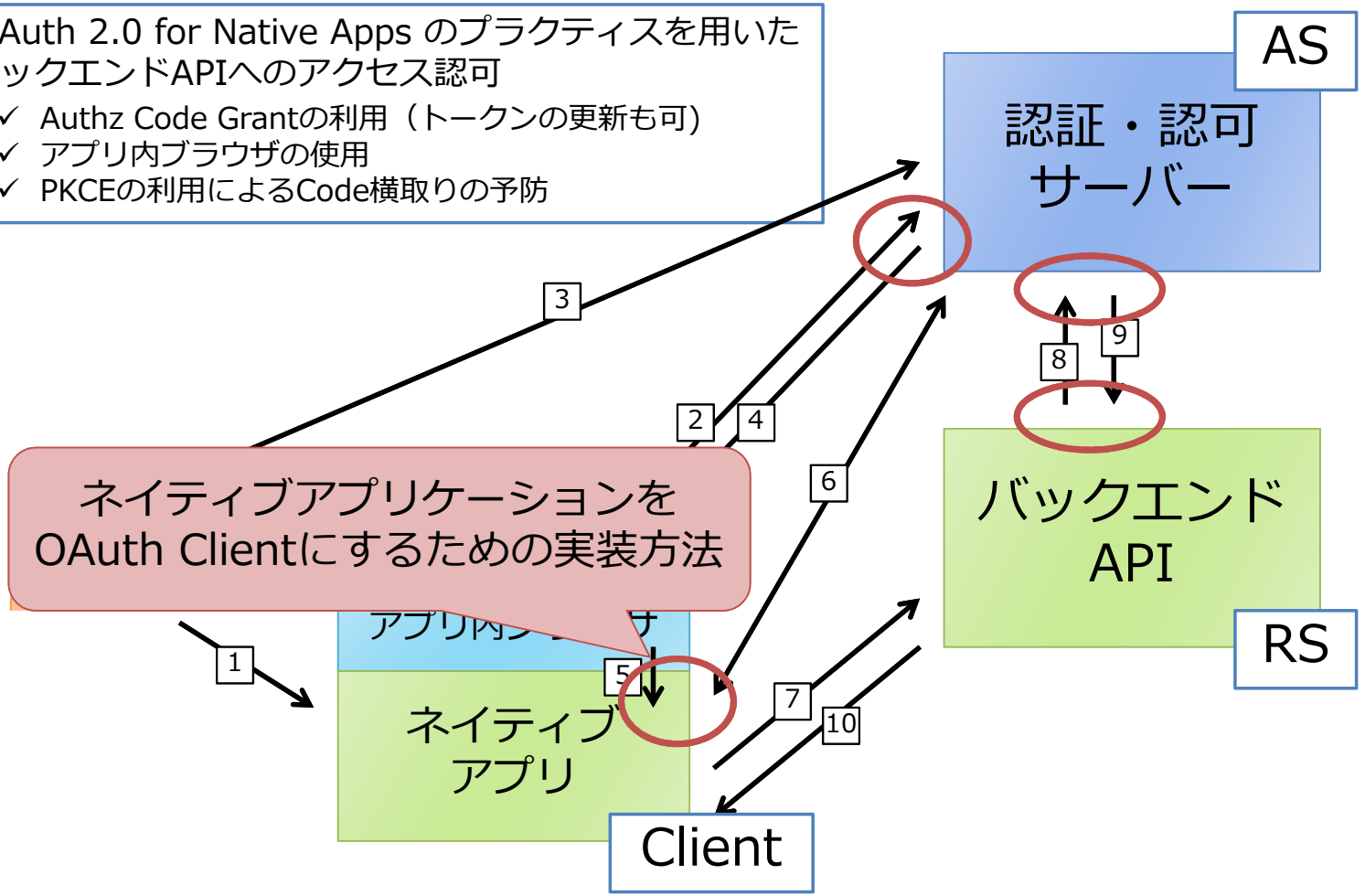
バックエンドAPIの振る舞いは、モバイルウェブ、ネイティブアプリで共通。

ネイティブアプリでのバックエンドAPIアクセス認可

	従業員向けサービス	顧客向けサービス
外部サービスの利用 SaaSが提供するモバイル向けアプリを利用	○	—
モバイルウェブの開発と提供	○	○
ネイティブアプリの開発・提供	○	◎
サードパーティとの連携 APIを提供し、サードパーティがアプリを開発・提供	—	○

OAuth 2.0 for Native Apps のプラクティスを用いたバックエンドAPIへのアクセス認可

- ✓ Authz Code Grantの利用（トークンの更新も可）
- ✓ アプリ内ブラウザの使用
- ✓ PKCEの利用によるCode横取りの予防



1. ネイティブアプリを起動、ログイン
2. アプリ内ブラウザを使いOAuth認可を要求（PKCEを併用）
3. ユーザー認証を実行
4. アクセストークンを取得するためのCodeを返却
5. アプリ内ブラウザからネイティブアプリにCodeを返却
6. Codeをアクセストークンに交換（PKCEを併用）
7. アクセストークンをつけてバックエンドAPIにアクセス
8. ~9. アクセストークンの情報を確認（ユーザー、Scope、有効期限、など）
10. バックエンドAPIが情報を使ってネイティブアプリが動作

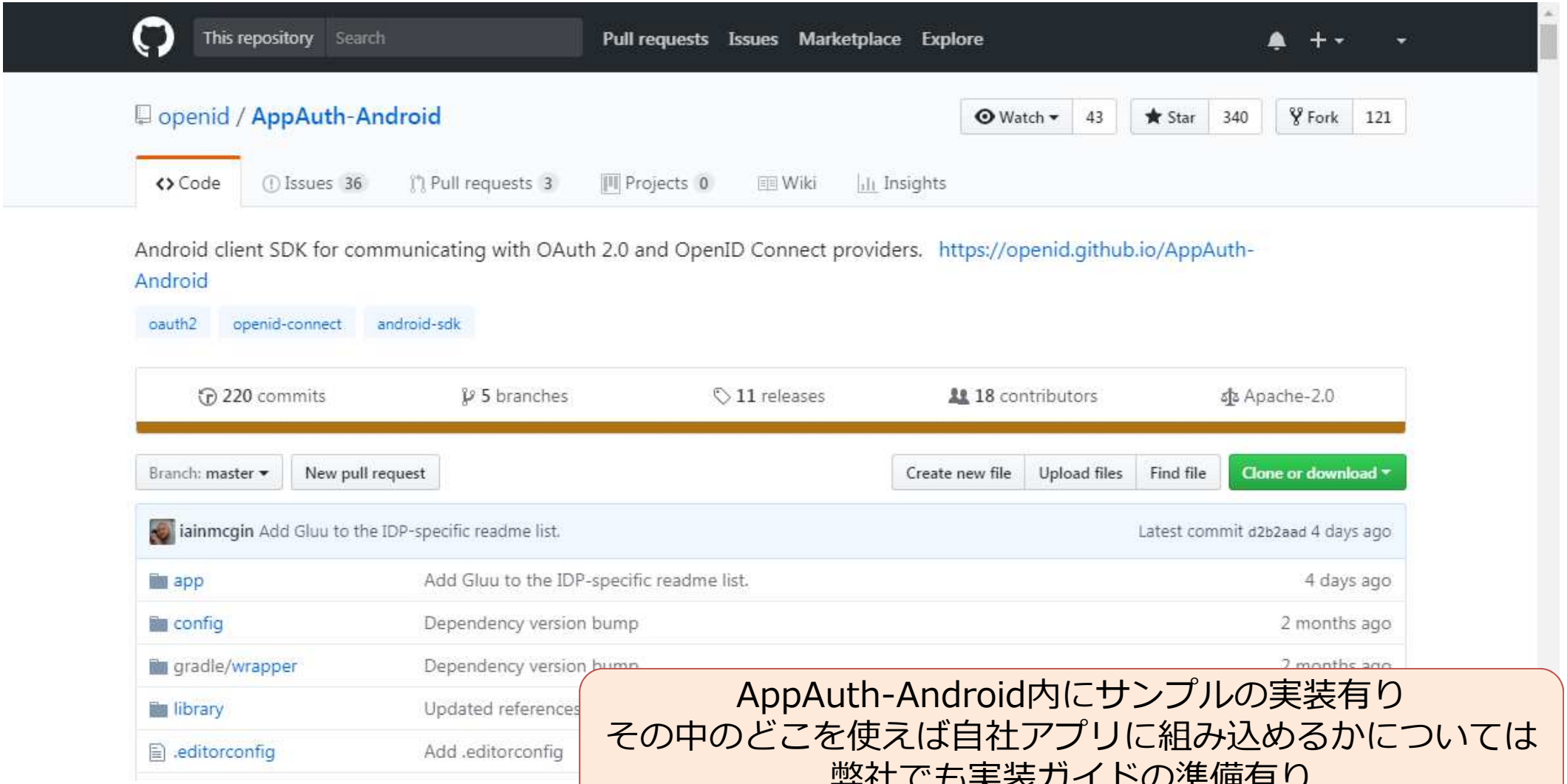
バックエンドAPIの振る舞いは、モバイルウェブ、ネイティブアプリで共通。

© 2017 OGIS-RI Co., Ltd.

AppAuth-Android
修正するファイルリスト

ファイル名	説明
AuthStateManager.java	ユーザの認可状態を管理するライブラリ。 ライブラリの入手先はAppAuth-Android
build.gradle	Android Studioのパッケージに同梱されているJava環境におけるビルドシステム。
MainActivity.java	認可サービスへ接続するための設定をして、 認可リクエストを送信する画面。
TokenActivity.java	認可レスポンスを受け取り、認可コードとトークンを交換する。また取得したAccess Tokenを使用してAPIにアクセスする画面。

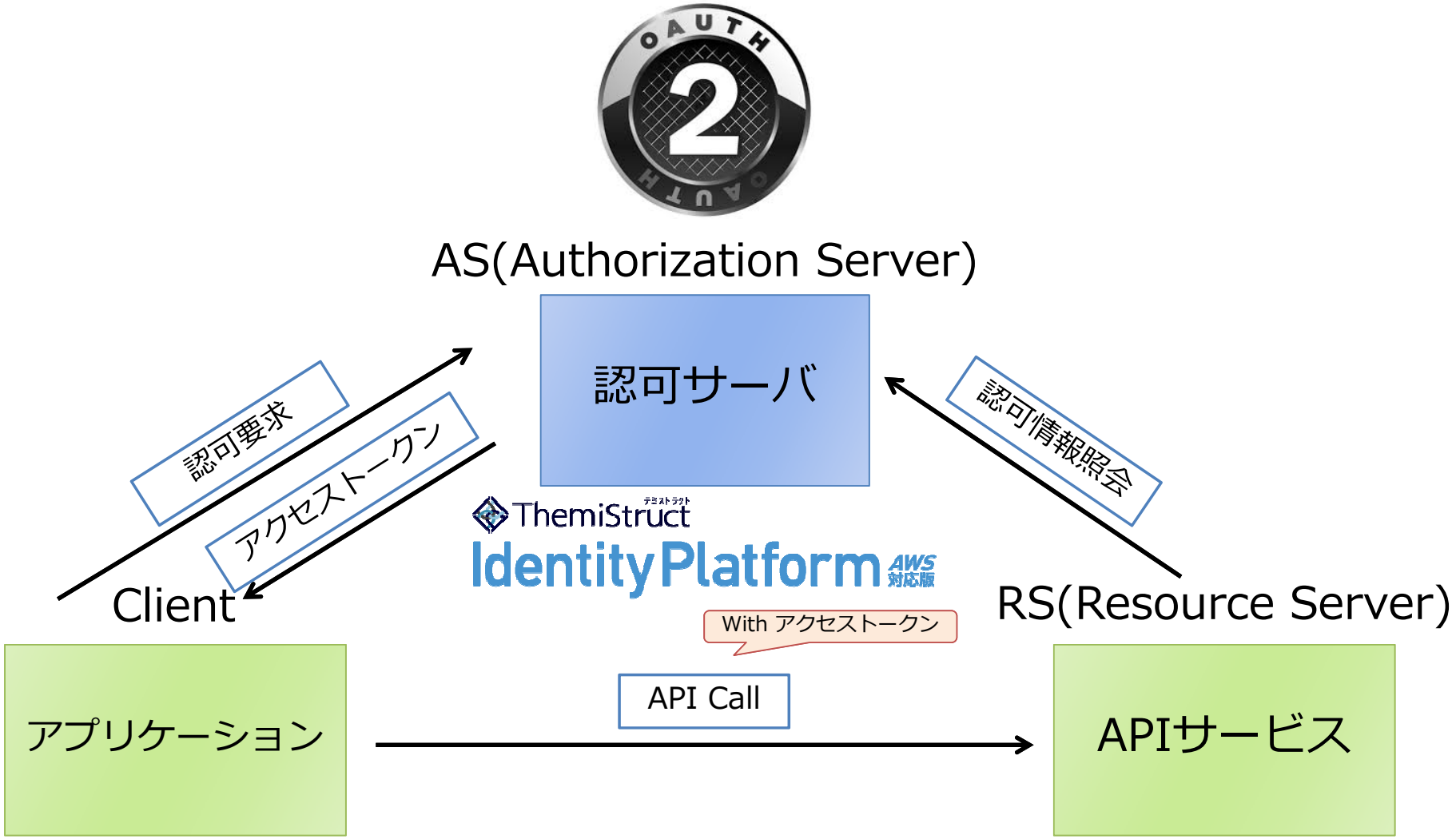
AppAuth-Android: https://github.com/openid/AppAuth-Android

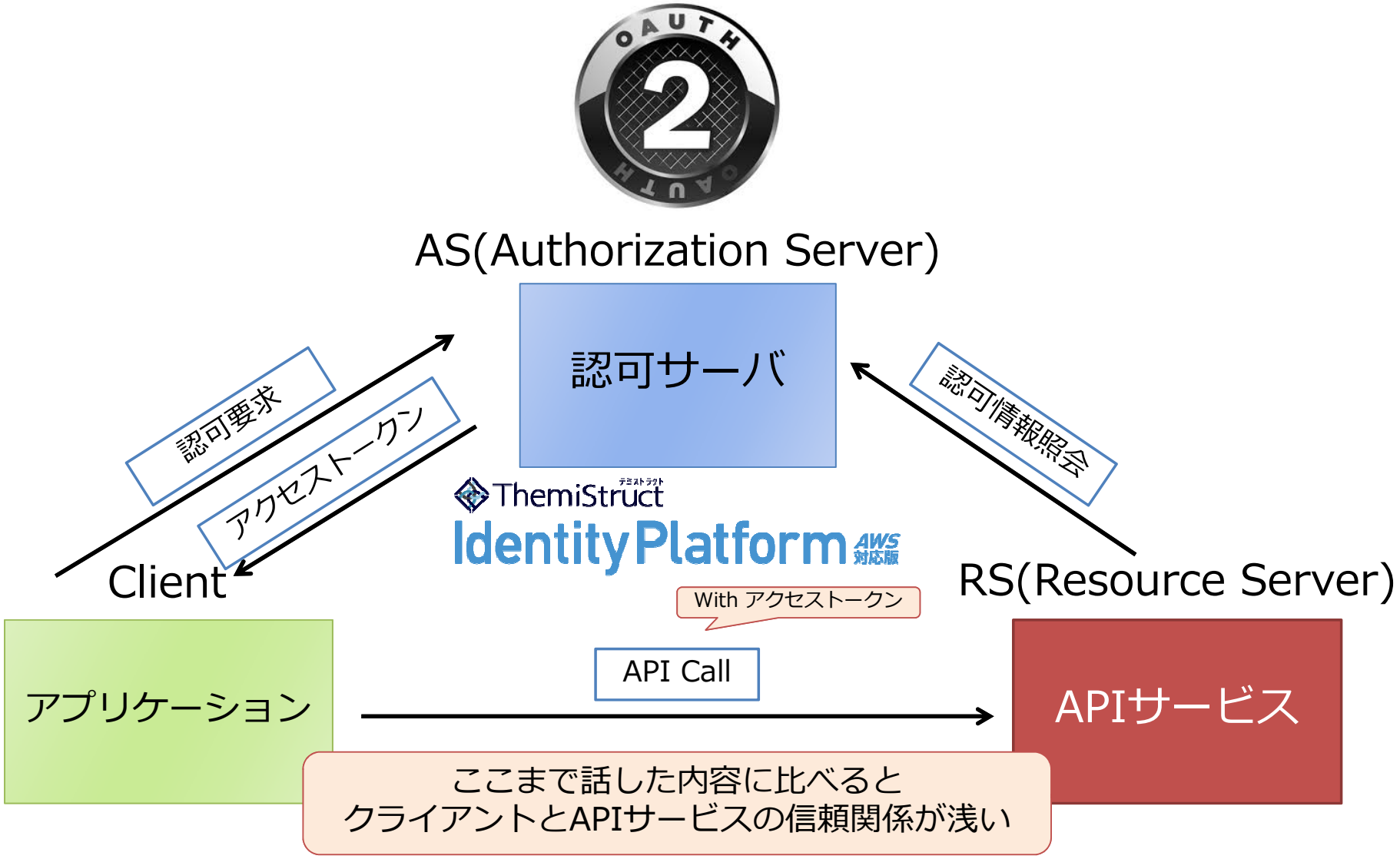


AppAuth-Android内にサンプルの実装有り
その中のどこを使えば自社アプリに組み込めるかについては
弊社でも実装ガイドの準備有り

認証・アクセス認可の実装の考え方

	従業員向けサービス	顧客向けサービス
外部サービスの利用 SaaSが提供するモバイル向けアプリを利用	OpenID Connect や SAML を使ったSaaSへのSSO	—
モバイルウェブの開発と提供	OAuth 2.0 Implicit Grant を用いたバックエンドAPIへのアクセス認可	
ネイティブアプリの開発・提供	OAuth 2.0 for Native Apps のプラクティスを用いたバックエンドAPIへのアクセス認可	
サードパーティとの連携 APIを提供し、サードパーティがアプリを開発・提供	—	OAuth 2.0 を用いたAPI連携時認証･アクセス認可 Implicit, Authz Code, Authz Code+PKCE （OIDF FAPI WGの動向の考慮も必要）

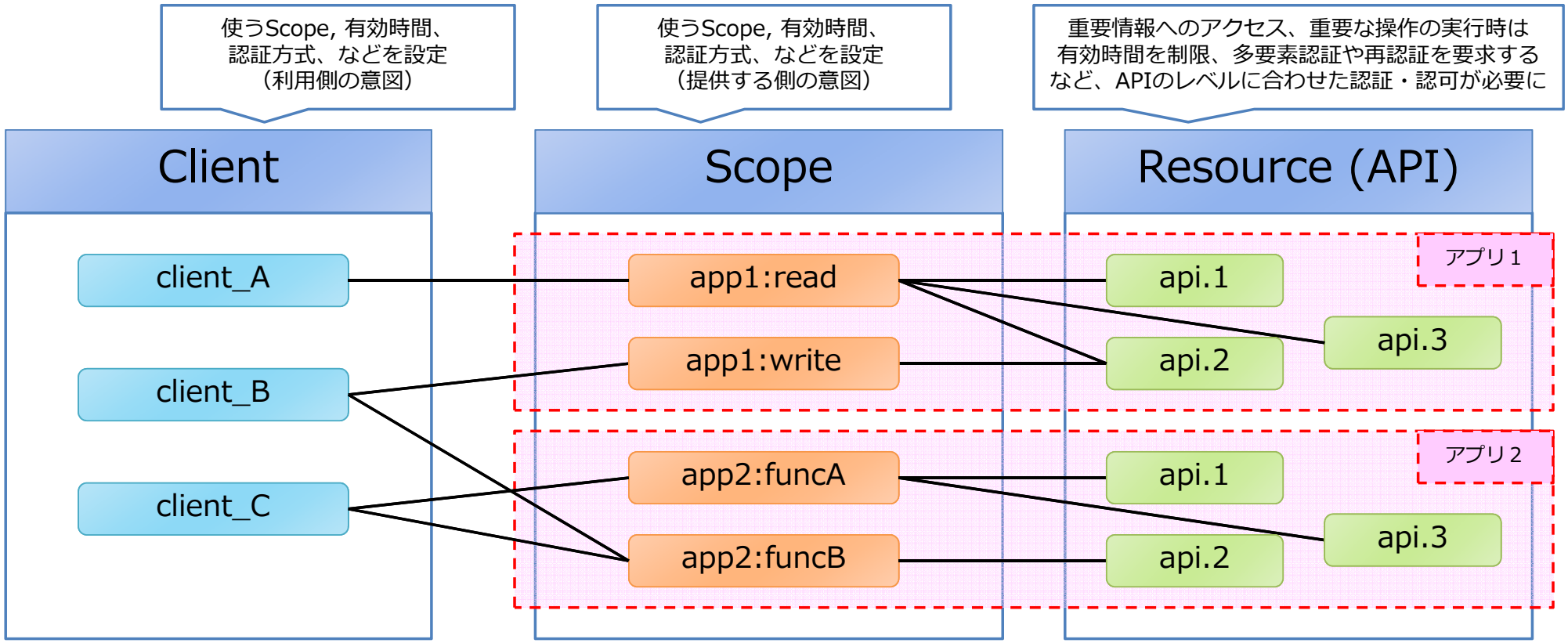

AS(Authorization Server)
認可サーバ
認可要求
アクセストークン
認可情報照会
ThemiStruct
IdentityPlatform AWS対応版
Client
RS(Resource Server)
With アクセストークン
API Call
アプリケーション
APIサービス


AS(Authorization Server)
認可サーバ
認可要求
アクセストークン
認可情報照会
ThemiStruct
Identity Platform
AWS対応版
Client
RS(Resource Server)
With アクセストークン
API Call
アプリケーション
APIサービス
ここまで話した内容に比べると
クライアントとAPIサービスの信頼関係が浅い

異なるレベルのAPI提供に対応できる API連携認証システム機能の必要性

	従業員向けサービス	顧客向けサービス
外部サービスの利用 SaaSが提供するモバイル向けアプリを利用	○	—
モバイルウェブの開発と提供	○	○
ネイティブアプリの開発・提供	○	◎
サードパーティとの連携 APIを提供し、サードパーティがアプリを開発・提供	—	○

API提供が進むと提供側の意図と矛盾が生じない認証・認可が課題となる



ClientとRSの要求によって発生する矛盾

スケジュール管理APIを使ってスケジュール管理アプリを作りたい

・一回認証したら長時間使いたい。
・いろんなAPIを呼び出したい

アプリケーション

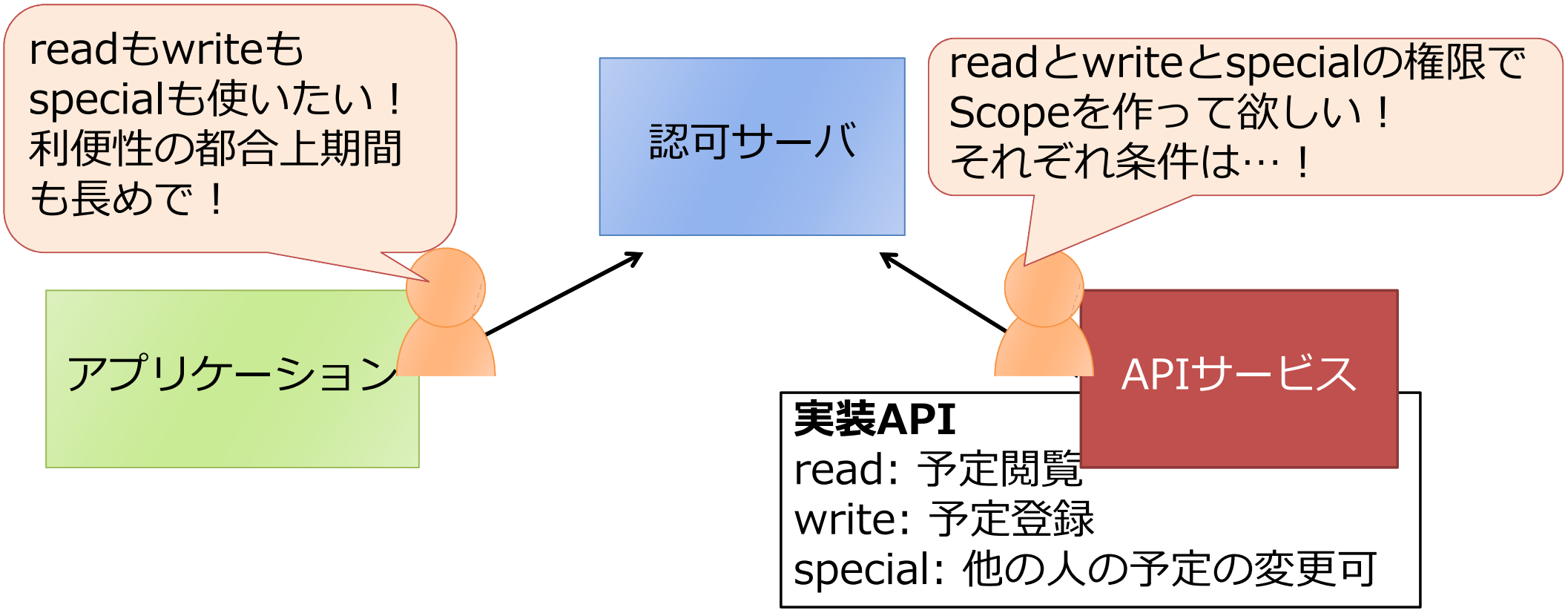
・閲覧は長時間できても良い
・他の人の予定を変更する場合は一定期間で再度認可を行いたい

APIサービス

実装API
read: 予定閲覧
write: 予定登録
special: 他の人の予定の変更可

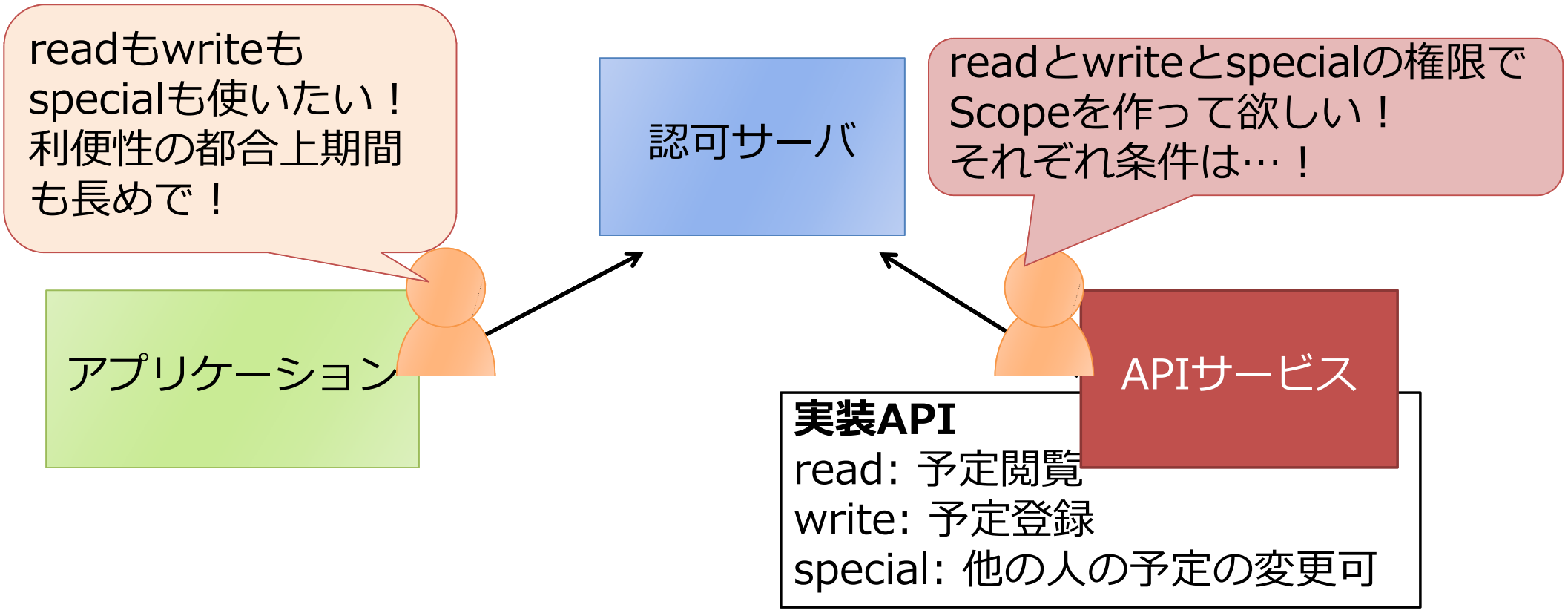
ClientとRSの要求によって発生する矛盾

スケジュール管理APIを使ってスケジュール管理アプリを作りたい



ClientとRSの要求によって発生する矛盾

スケジュール管理APIを使ってスケジュール管理アプリを作りたい



Scopeの登録
Identity Platform管理画面

期限を長めにセット(1時間)

このスコープでは
リフレッシュトークン利用許可

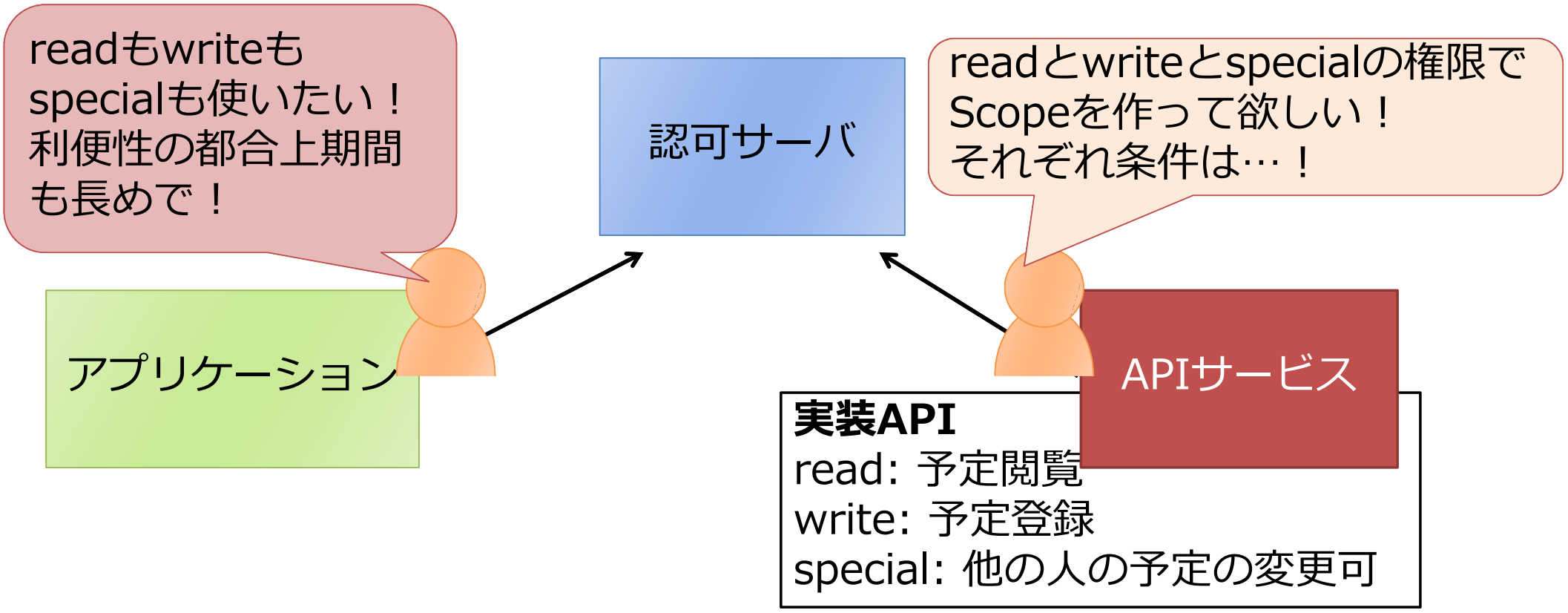
Scopeの登録
Identity Platform管理画面

期限を短めにセット(10分)

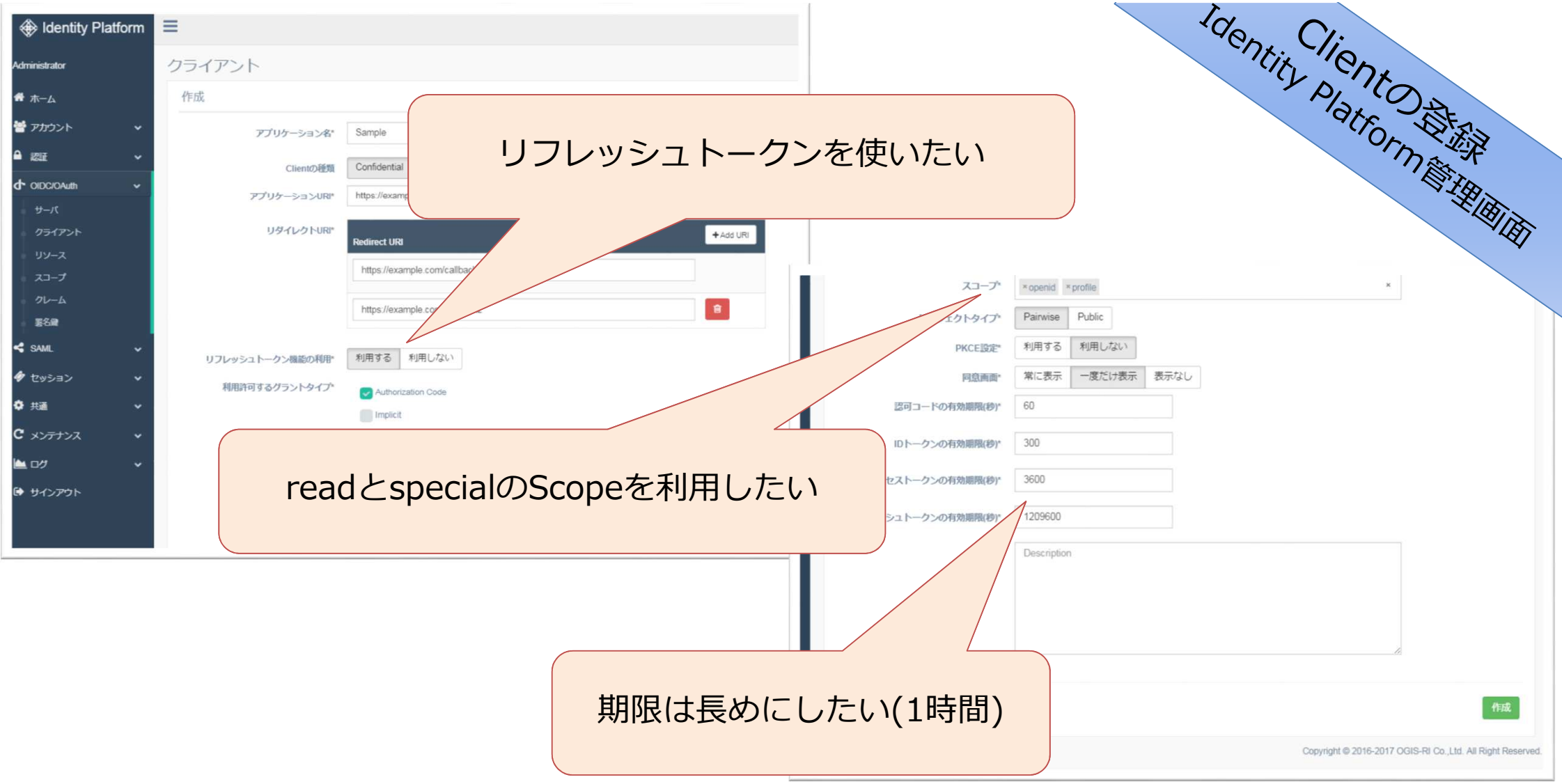
このスコープでは
リフレッシュトークンを使ってほしくない

ClientとRSの要求によって発生する矛盾

スケジュール管理APIを使ってスケジュール管理アプリを作りたい



Clientの登録
Identity Platform管理画面



リフレッシュトークンを使いたい

readとspecialのScopeを利用したい

期限は長めにしたい(1時間)

アクセストークン

ClientがreadのScopeを要求した際のトークンレスポンス

```
{
  "access_token":"SampleAccessToken",
  "token_type":"bearer",
  "expires_in":"3600",
  "refresh_token":"SampleRefreshToken"
}
```

ClientがreadとspecialのScopeを要求した際のトークンレスポンス

```
{
  "access_token":"SampleAccessToken",
  "token_type":"bearer",
  "expires_in":"600"
}
```

おわりに

◘認可処理の中身の具体イメージ

- ➢連携されている情報はどのようなものなのか
- ➢実装の簡単なイメージ

◘課題に対する実現の具体イメージ

- ➢モバイルウェブからバックエンドAPIを叩く際の認可
- ➢ネイティブアプリからバックエンドAPIを叩く際の認可
- ➢サードパーティからAPIを叩く際の認可

© 2017 OGIS-RI Co., Ltd.

ご清聴ありがとうございました

ThemiStruct
テミストラクト

【お問い合わせ先】
株式会社オージス総研
TEL: 03-6712-1201 / 06-6871-7998
mail: info@ogis-ri.co.jp

OGIS-RI

© 2017 OGIS-RI Co., Ltd.