

今更聞けない OAuth2.0

presented by @white_aspara25

自己紹介

- @white_aspara25
- istyle.inc エンジニア（自称）
- ここ1年は ハノイ@ベトナム で Bridgeエンジニアとして働いてました

話す内容

1. OAuth ってなに？
2. OAuth2.0 システム仕様
3. セキュリティ対策

話さない内容

1. OAuth1.0 の仕様
2. XAuth の仕様
3. OpenID Connect

話す内容

1. OAuth ってなに？
2. システム仕様
3. セキュリティ対策

おーす
OAuth ?

OAuth って？

HTTP 上で 認可

を行うためのプロトコル

※認証ではなく、認可です

OAuth って？

HTTP 上で 認可

を行うためのプロトコル

※認証ではなく、認可です

認証 or 認可 ?

認証？

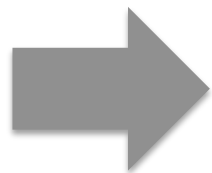
- 英: Authentication



「認証(にんしょう)とは、何かによって、対象の正当性を確認する行為を指す」

「相手認証とは、ある人が確かに本人であると納得させる事をいう」

by Wikipedia



「誰？」

を確認するのが目的

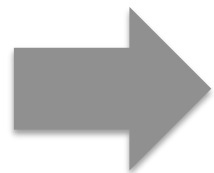
認可？

- 英: Authorization



「リソースのアクセス権を指定する機能であり、(中略)特にアクセス制御と関係性が深い」

by Wikipedia



「何ができる？」

を確認するのが目的

OAuth = 認可

の Protokol です。
「誰か？」を確認する目的のものではありません。

OAuth = 認可

の Protokol です。

「誰か？」を確認する目的のものではありません。

= OpenID Connect (今回は説明しません)

OAuth って？

HTTP 上で 認可

を行うためのプロトコル

※認証ではなく、認可です

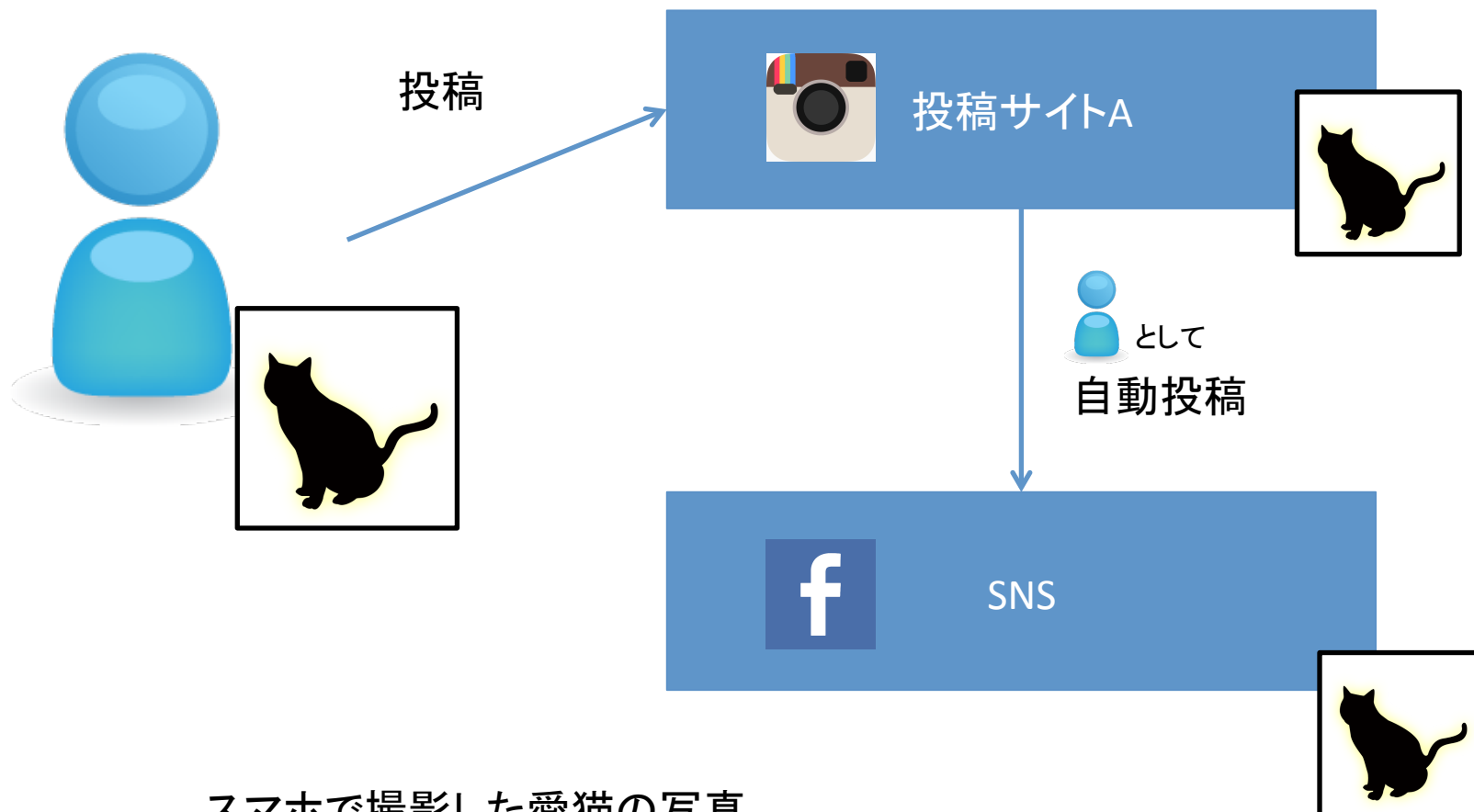
OAuth って？

HTTP 上で 何が できる？

を確認するためのプロトコル

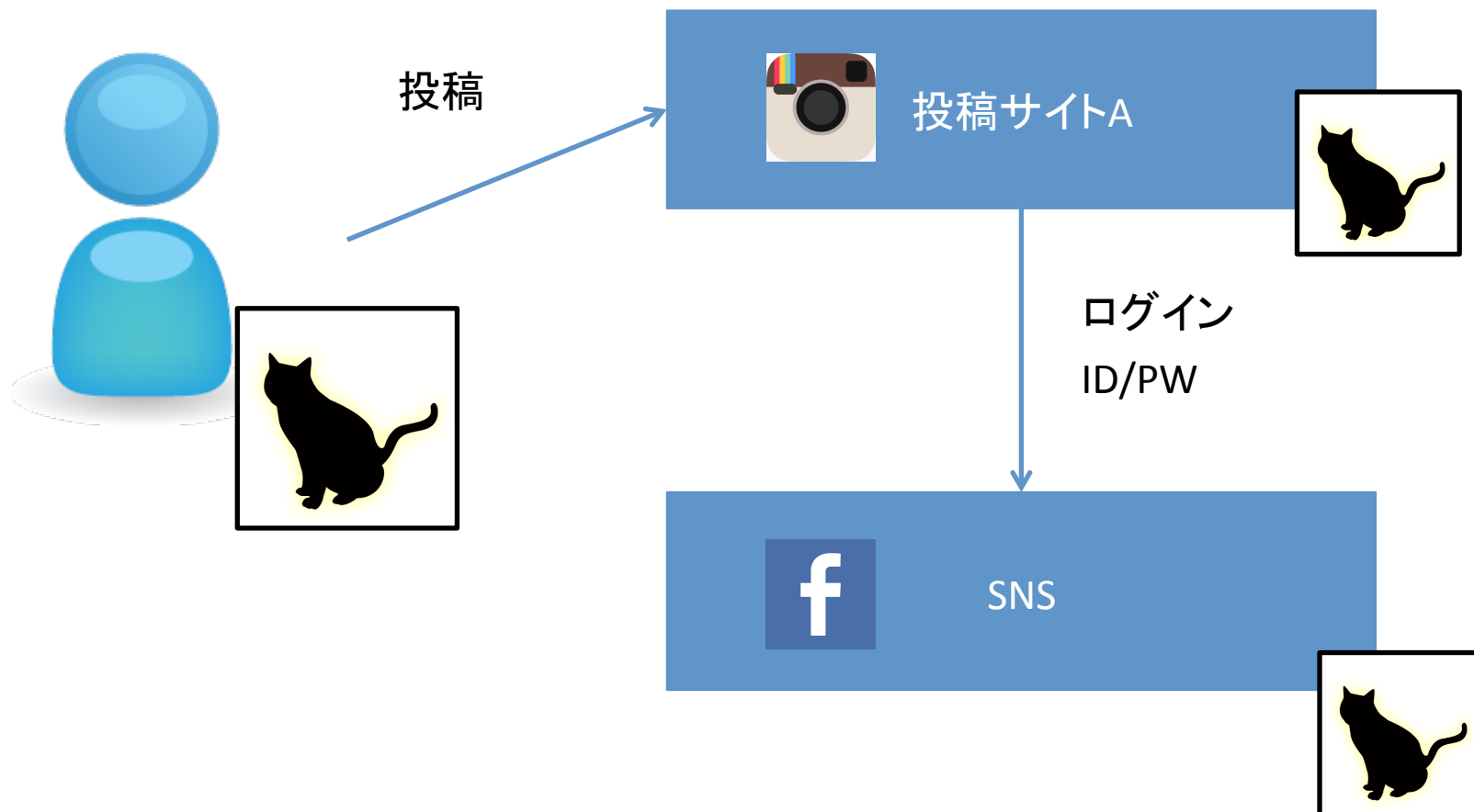
※認証ではなく、認可です

例



スマホで撮影した愛猫の写真
サイトAに投稿したら、自動でSNSにも投稿したい！

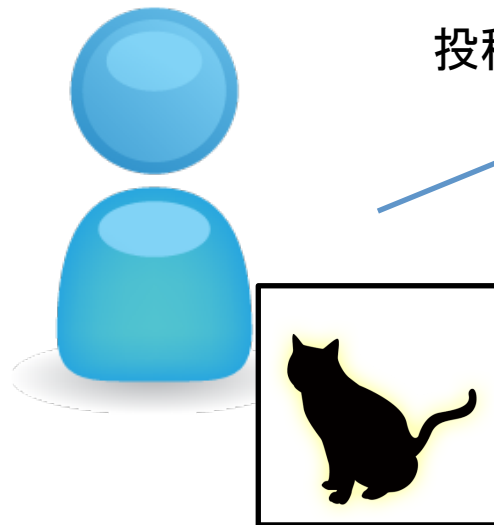
一番簡単な方法



一番簡単



の ID/PW を保持している



投稿



投稿サイトA



ログイン
ID/PW



SNS



But...

それ、アカンやつや

- サイトA はユーザの許可がなくても SNS の情報が見れてしまう
- SNS のパスワードを変更
=> サイトAのパスワードも変更が必要
- ID/PW漏洩のリスク

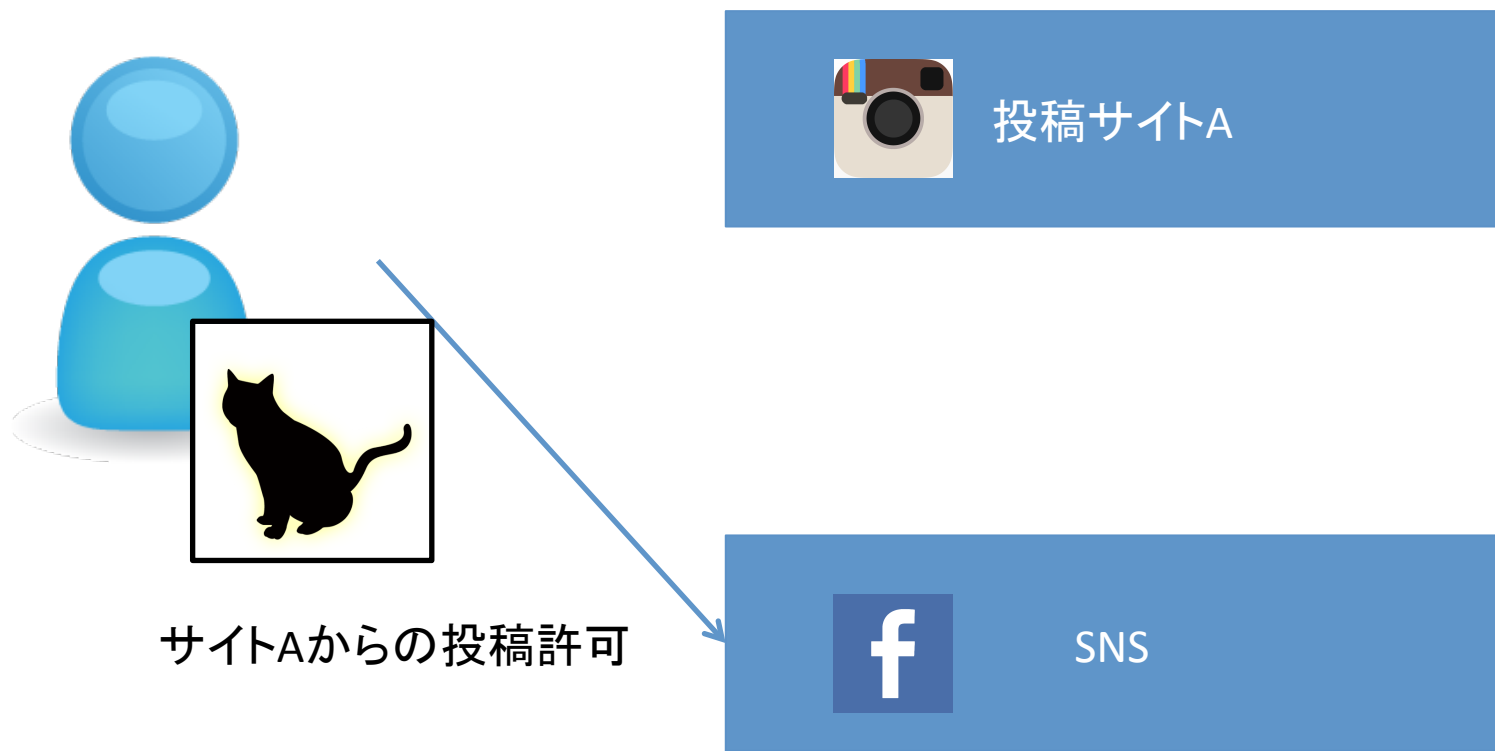
etc...

問題大有りです。

そこで

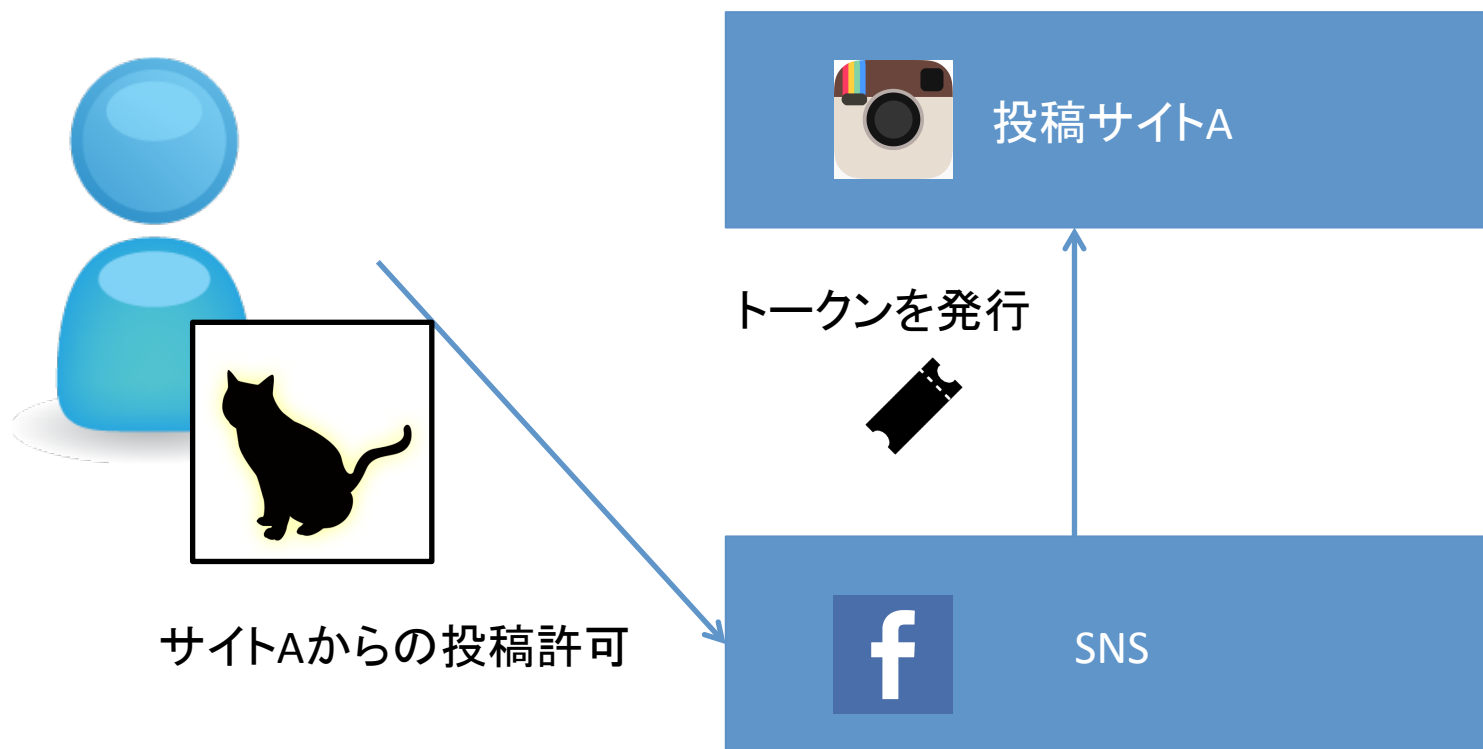
OAuth

OAuth を使った方法



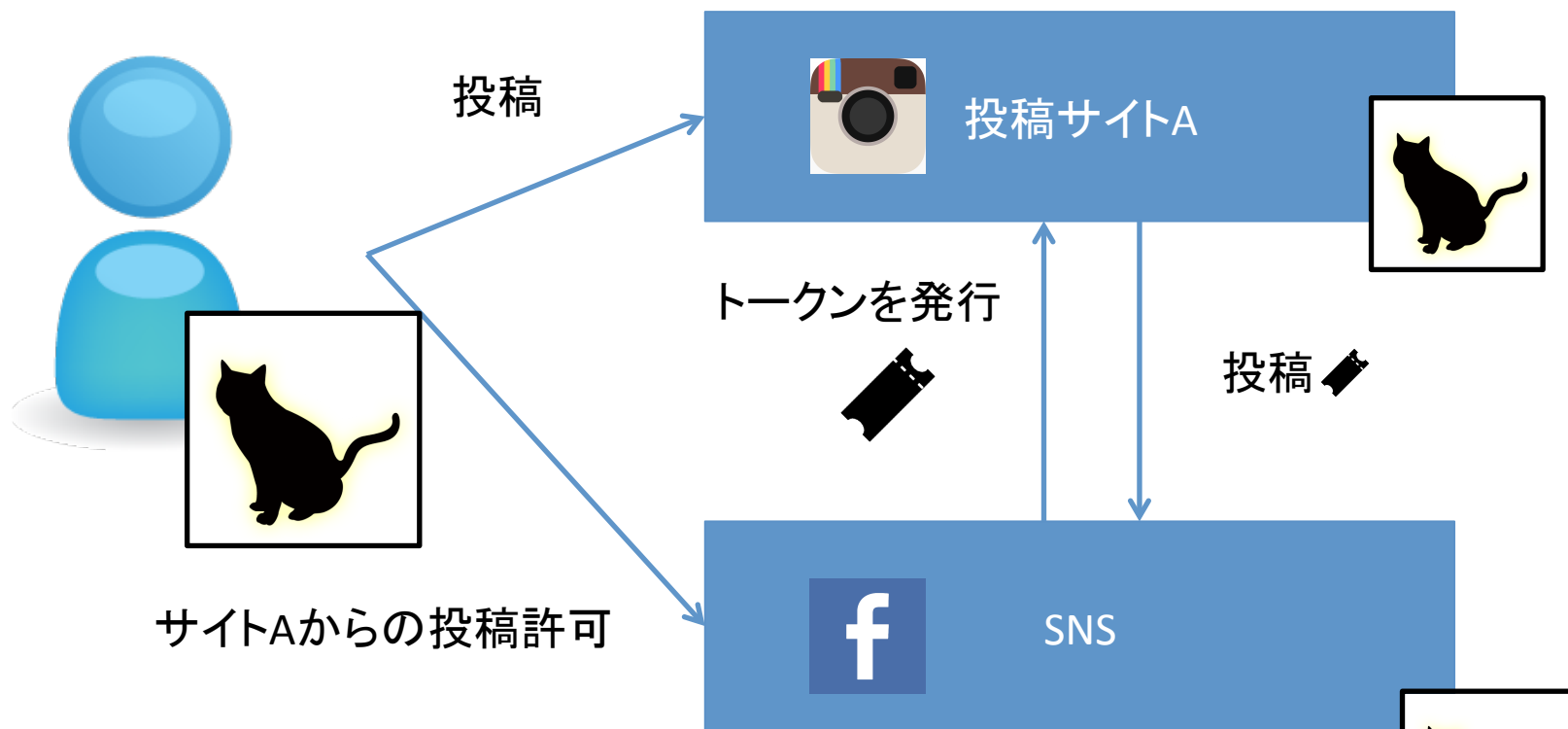
1. ユーザはSNSにサイトAからの投稿を許可(＝認可)

OAuth を使った方法



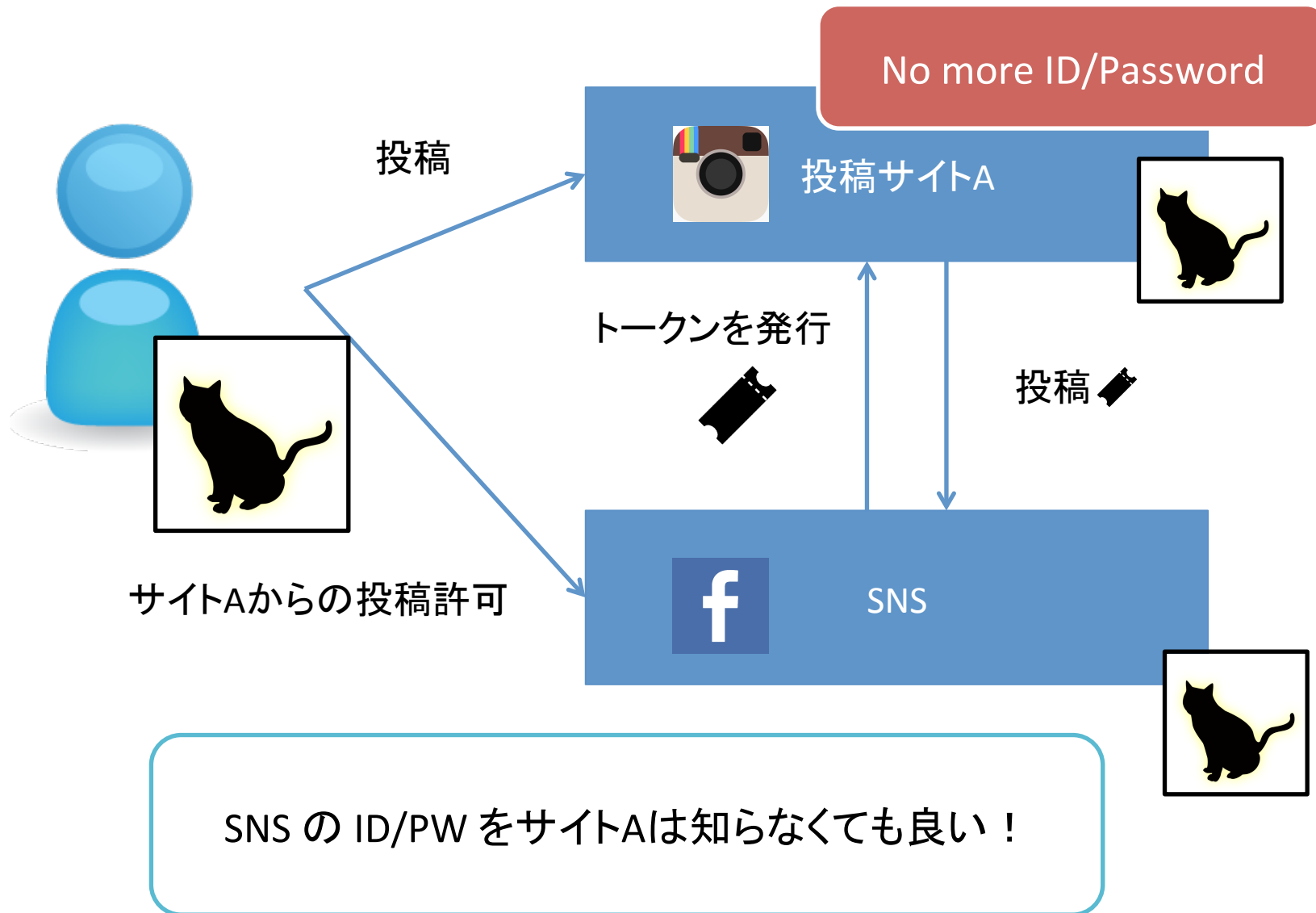
1. ユーザはSNSにサイトAからの投稿を許可(=認可)
2. ユーザの許可を受けてSNSはトークンを発行

OAuth を使った方法



1. ユーザはSNSにサイトAからの投稿を許可(=認可)
2. ユーザの許可を受けてSNSはトークンを発行
3. 投稿サイトAはトークンを使ってSNSに投稿

OAuth を使った方法



注意点



トークン

=

xxのサービス上で

△△の代わりに

〇〇ができる

を証明するもの

〇〇

=

投稿（メッセージ、写真、、）

閲覧（個人情報、友達、、）

注意点



トークン

=

xxのサービス上で
△△の代わりに

〇〇ができる

を証明するもの

〇〇

ただのチケット
= 本人じゃなくても使える
= 本人証明にならない

夏、、)

三、、)

注意点



=

xxのサービス上で
△△の代わりに

〇〇ができる

を証明するもの

トークン

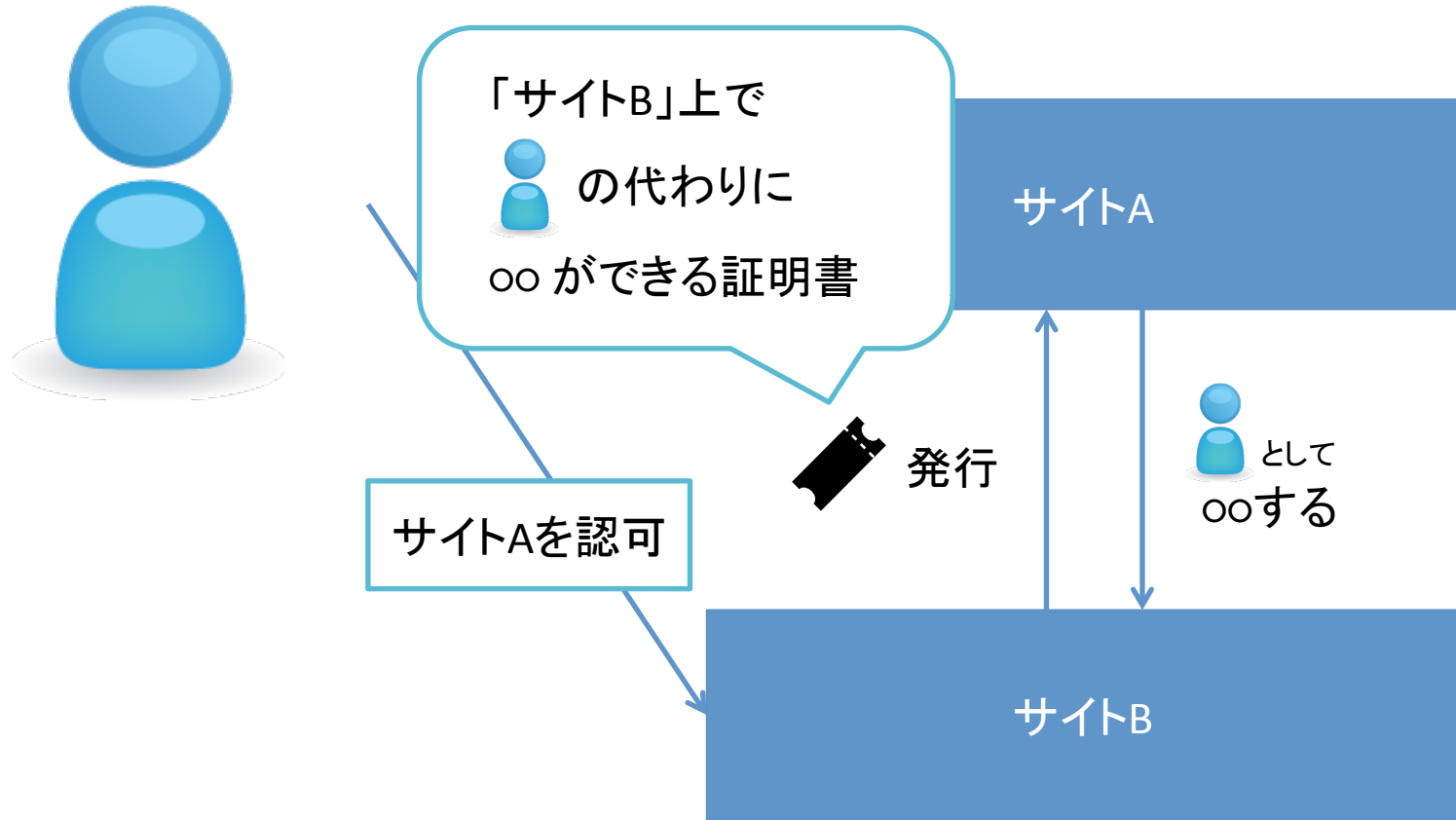
〇〇

用法用量を守って

正しくお使いください

、、)
、、)

まとめると



この一連のフローを仕様に落としたのが
OAuth

話す内容

1. OAuth ってなに？
2. OAuth2.0 システム仕様
3. セキュリティ対策

OAuth2.0

4種類のトークン  発行フロー

- Authorization Code Grant
- Implicit Grant
- Resource Owner Password Credentials Grant
- Client Credentials Grant

OAuth2.0

4種類のトークン  発行フロー

- Authorization Code Grant
- Implicit Grant
- Resource Owner Password Credentials Grant
- Client Credentials Grant

今回は取り扱いません

Authorization Code Grant

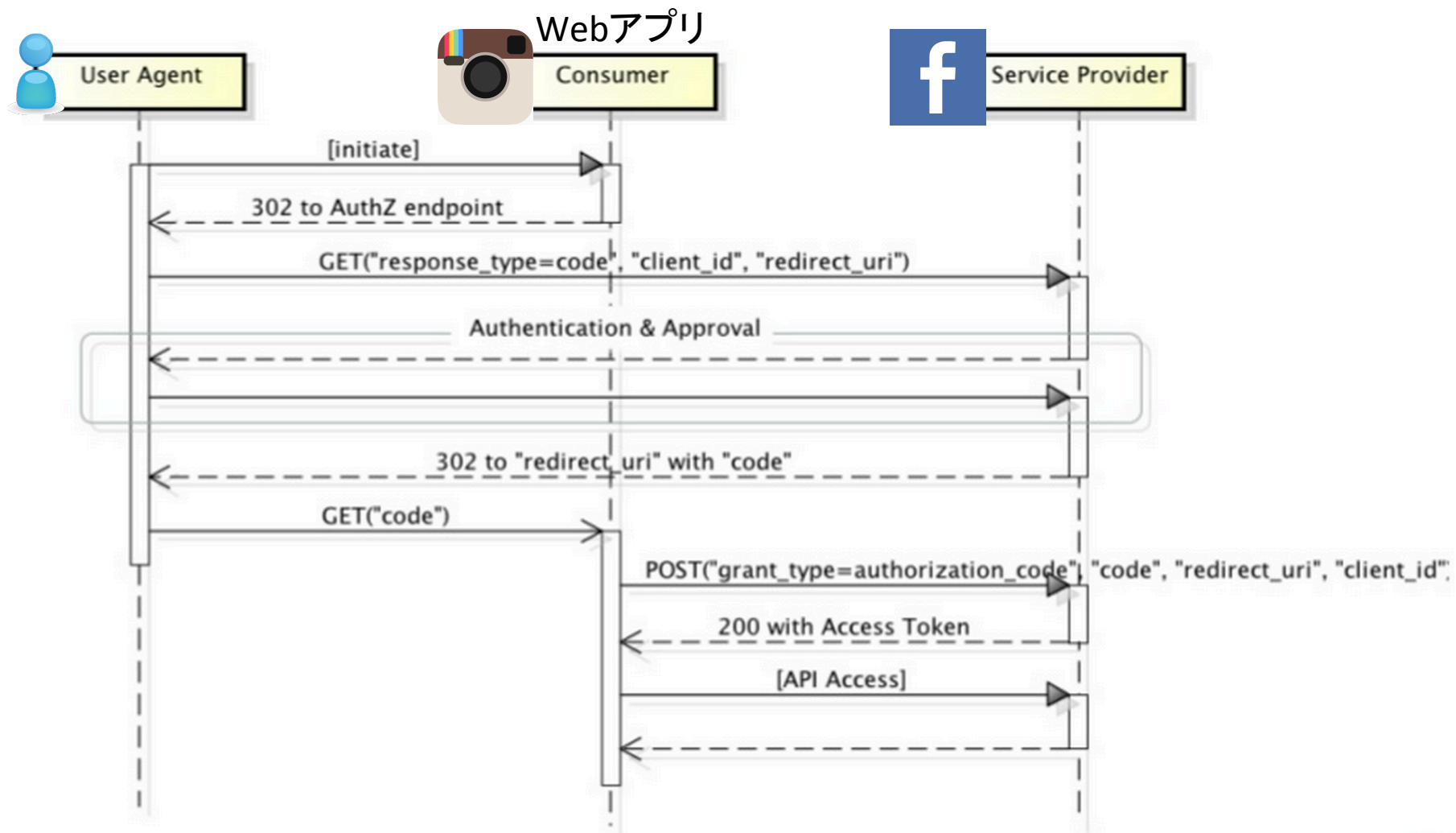
- 認可コードグラント、とも
- サーバ間通信のような、トークンを秘密保持できる場合に使われる
- 一般的な web アプリはコレ

Implicit Grant

- トークンを秘密保持できない場合に。
- スマホアプリや Javascript はコレ。

Authorization Code Grant

Authorization Code Grant 処理の流れ



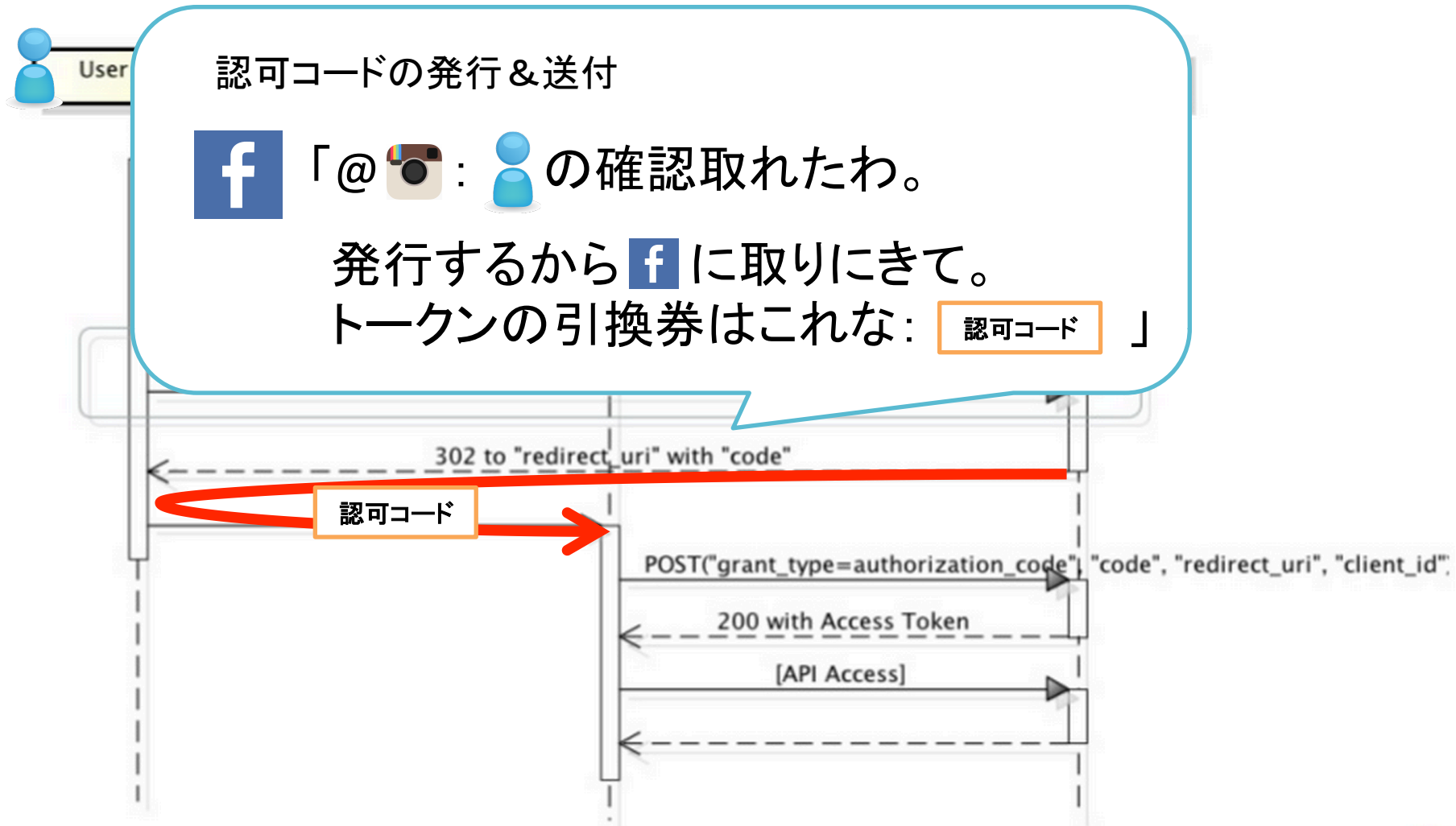
Authorization Code Grant 処理の流れ



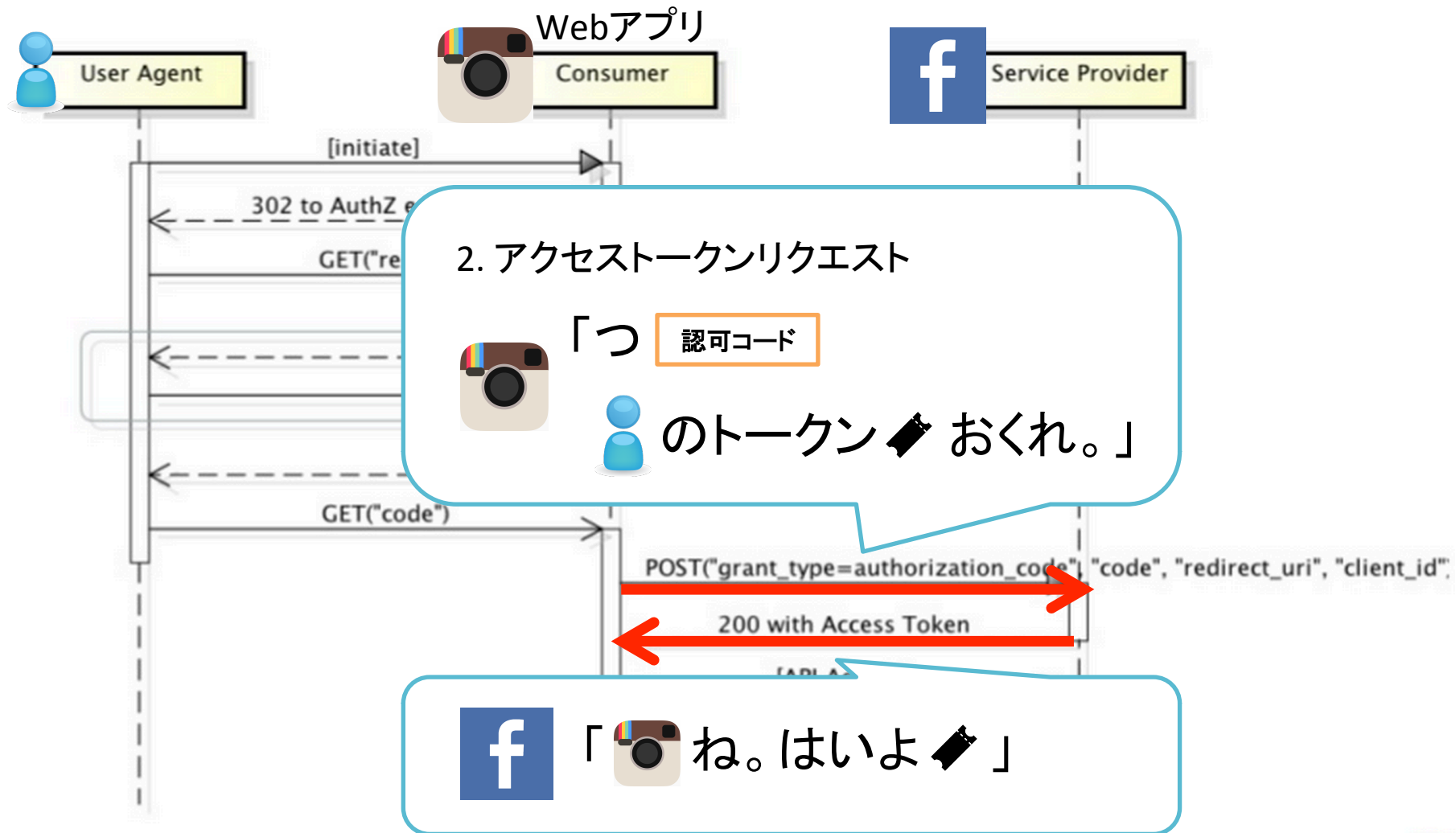
Authorization Code Grant 処理の流れ



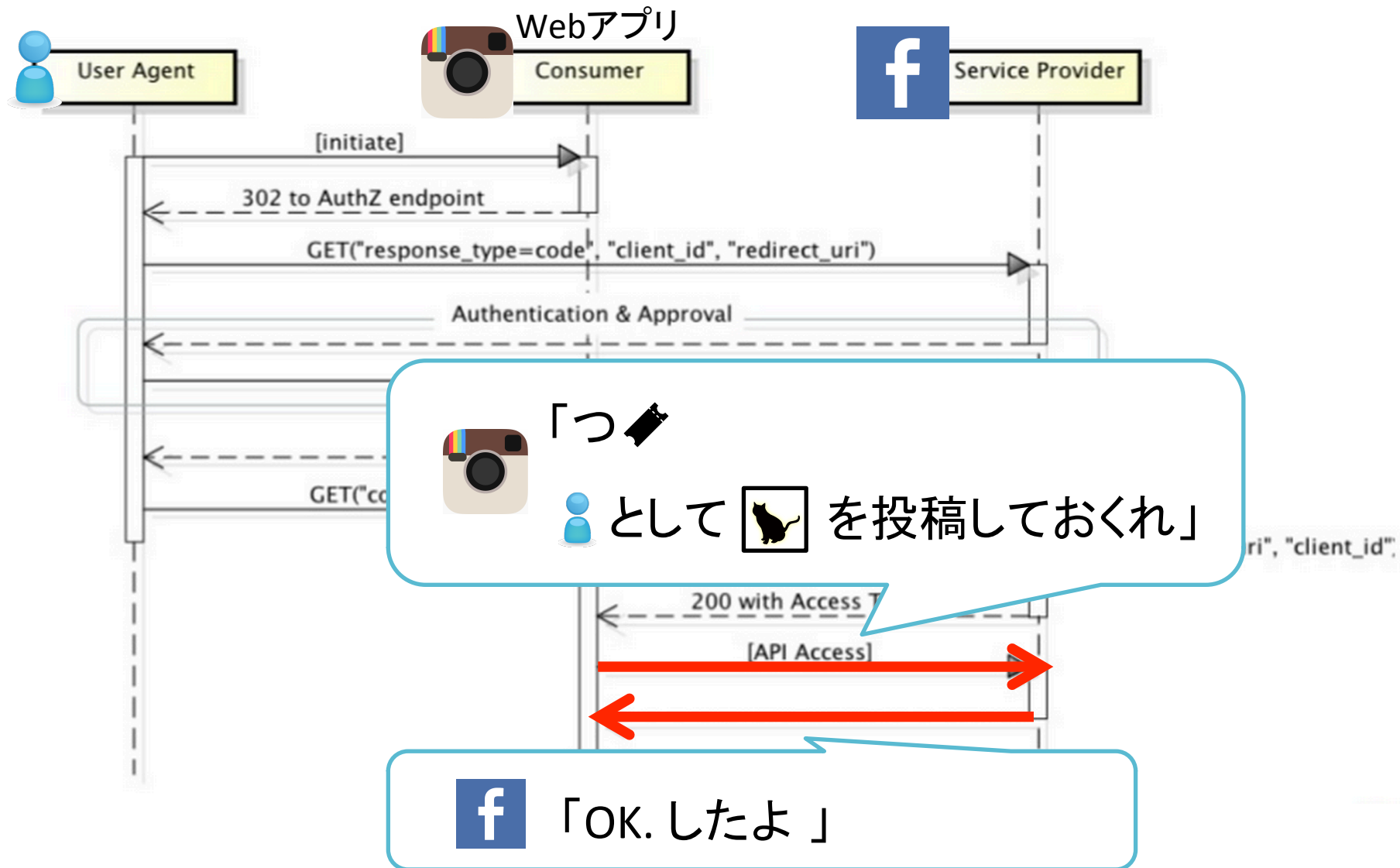
Authorization Code Grant 処理の流れ



Authorization Code Grant 処理の流れ



Authorization Code Grant 処理の流れ



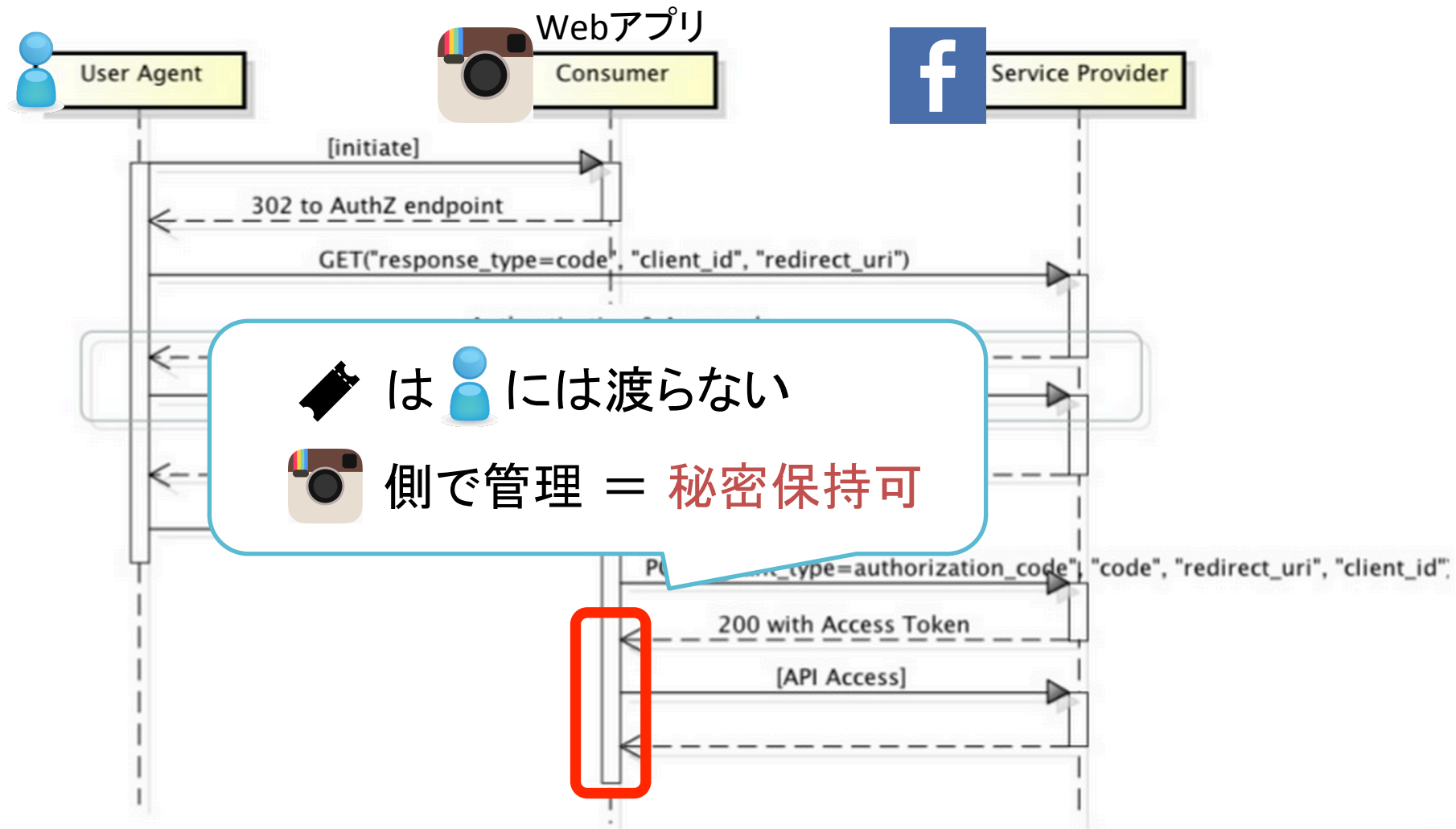
Authorization Code Grant

ポイント

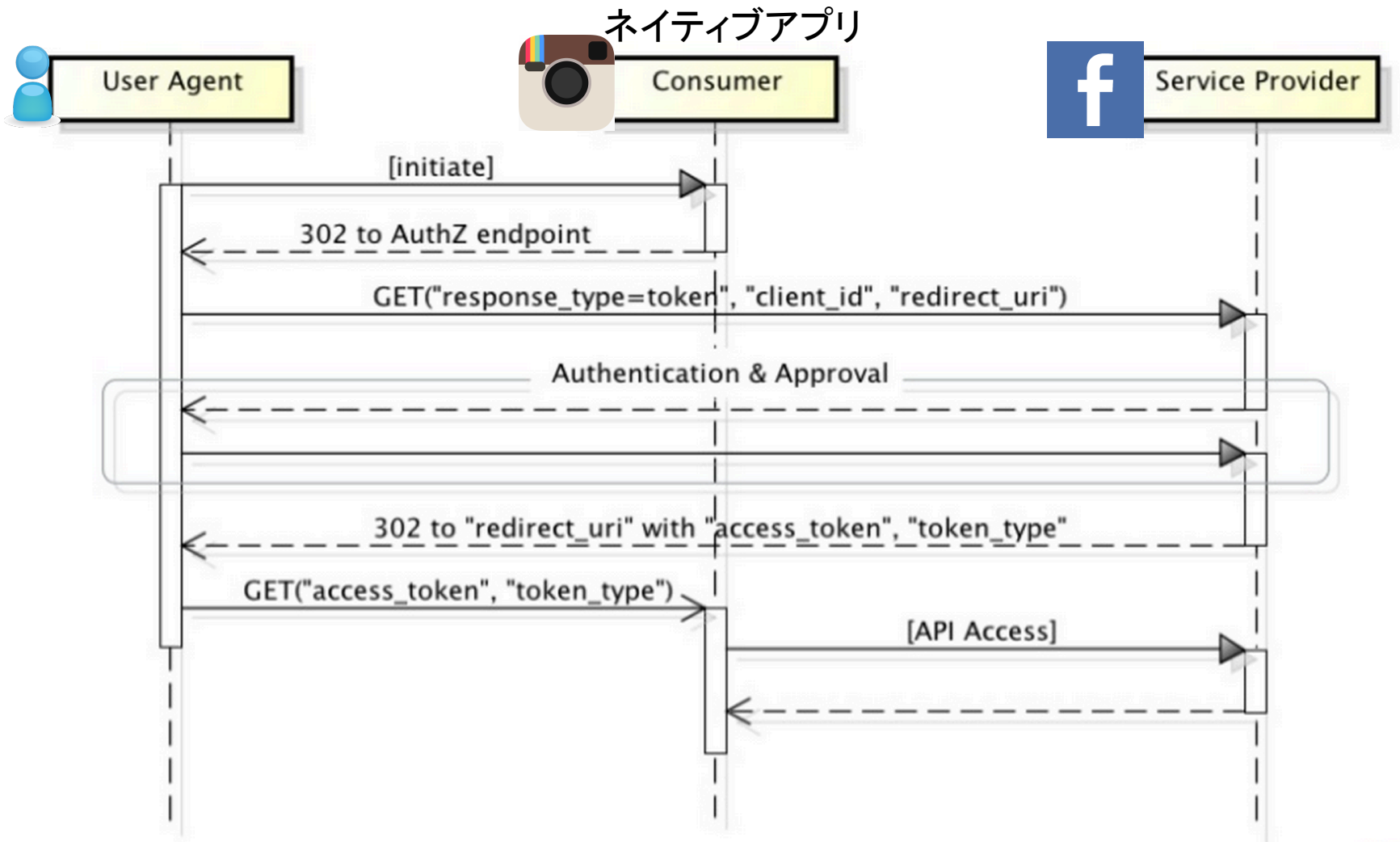
ポイント



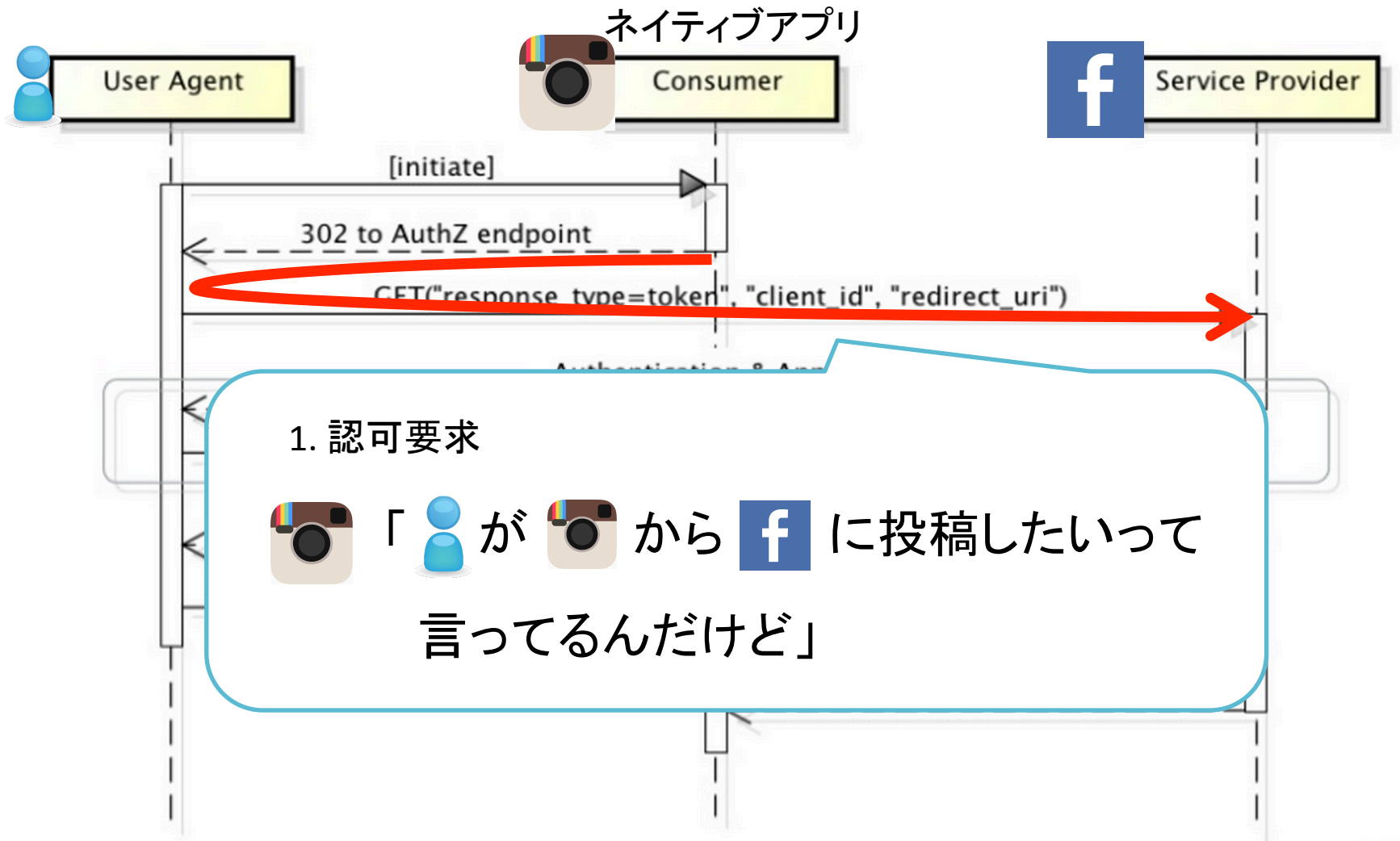
ポイント



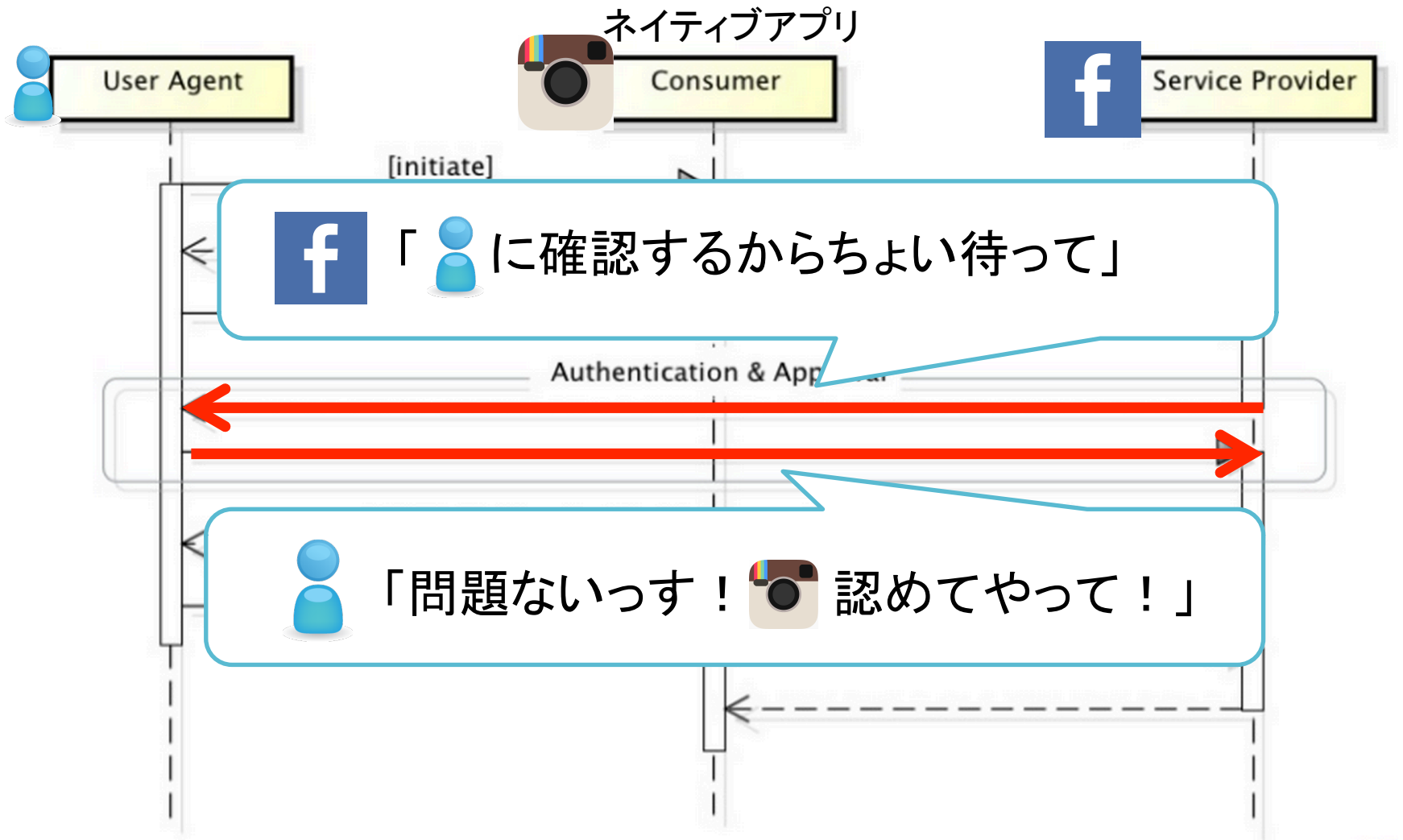
Implicit Grant 処理の流れ



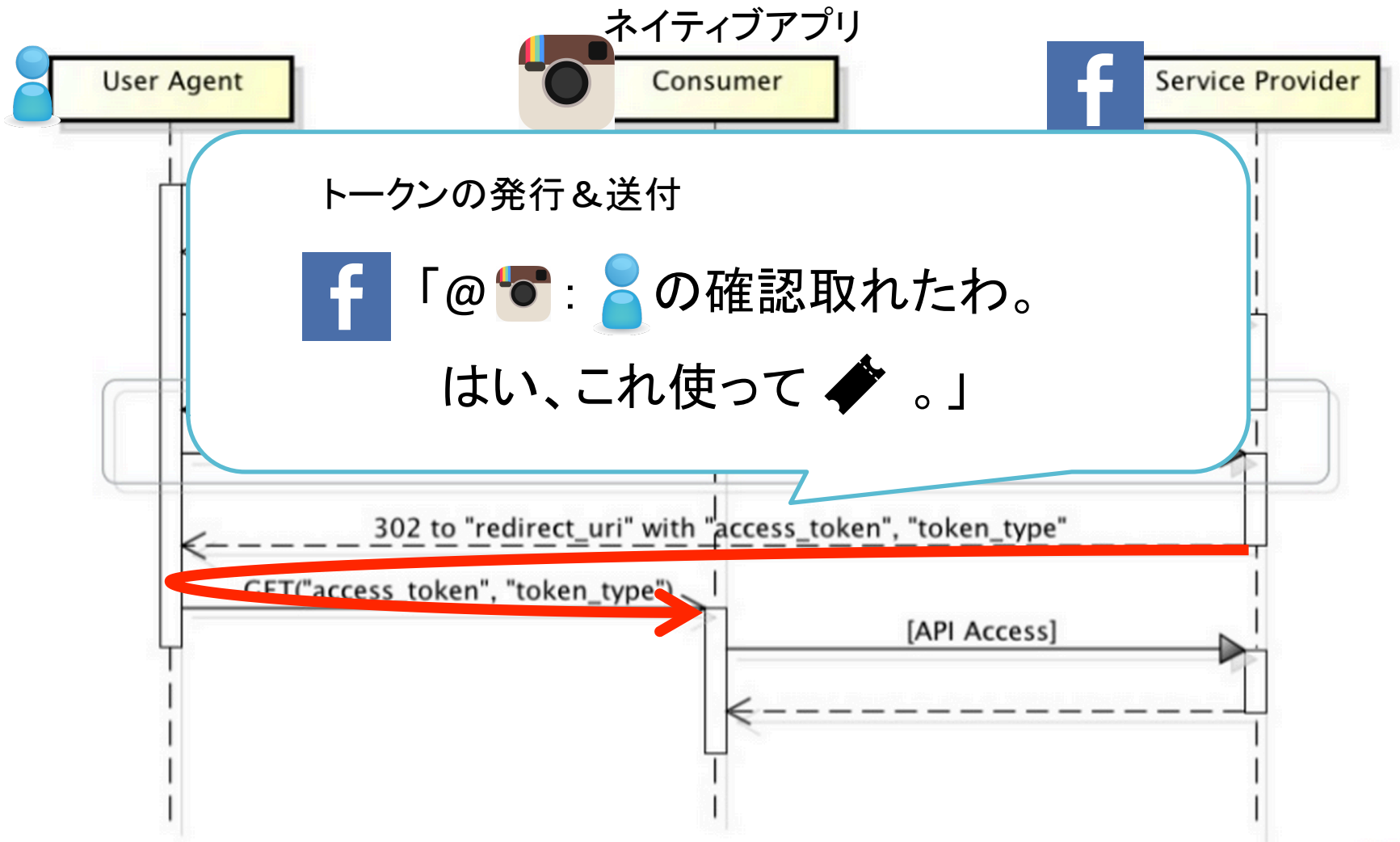
Implicit Grant 処理の流れ



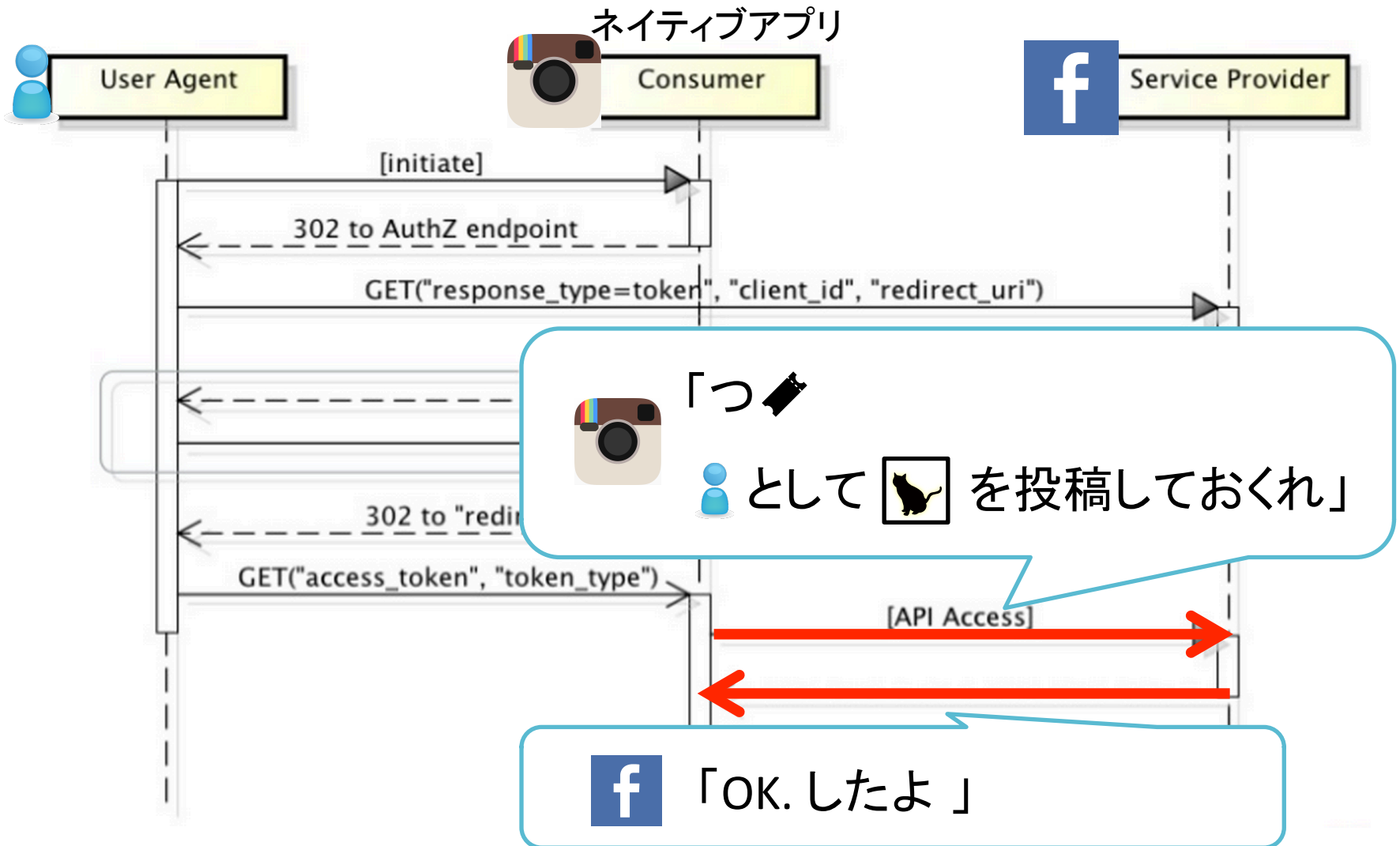
Implicit Grant 処理の流れ



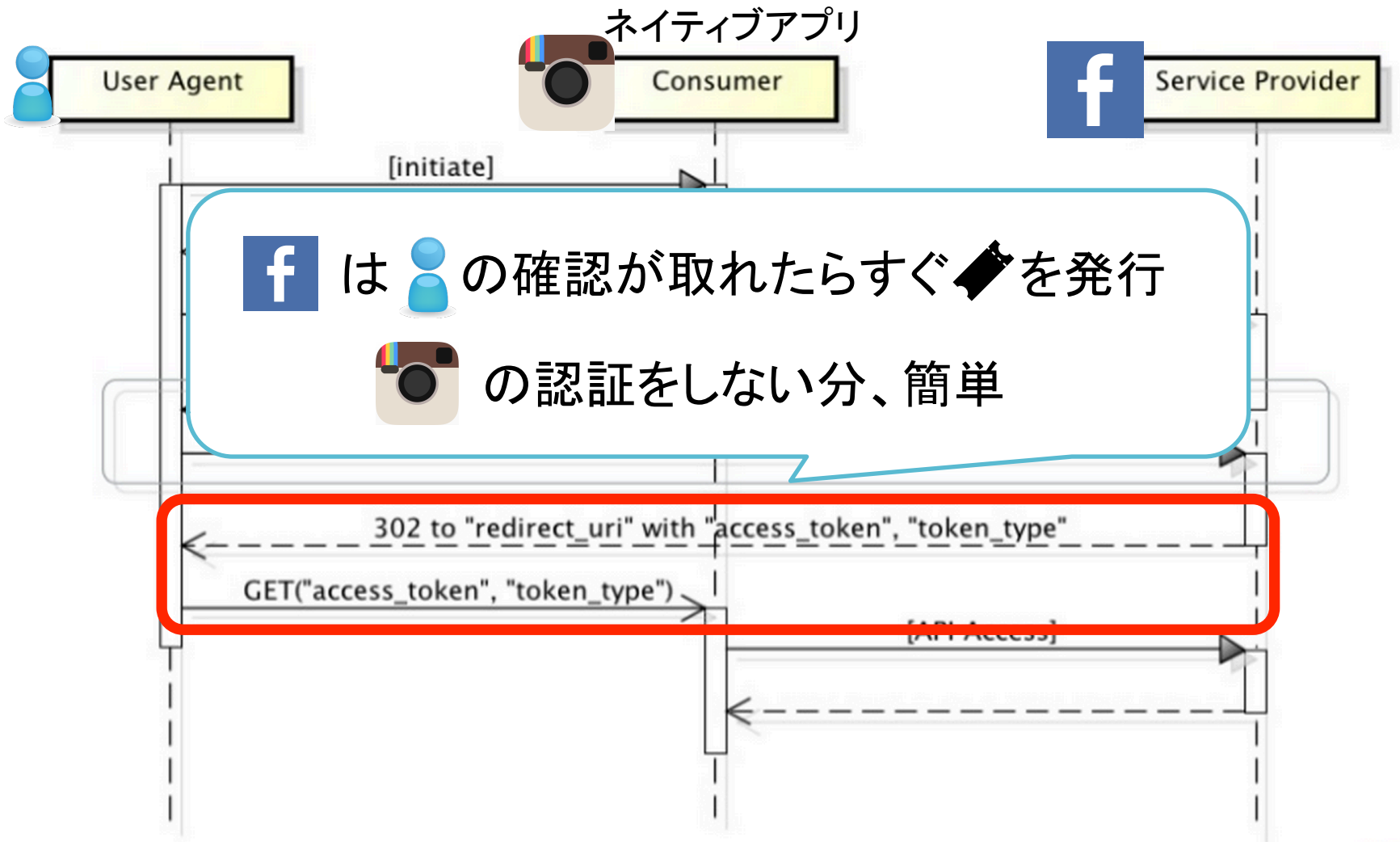
Implicit Grant 処理の流れ



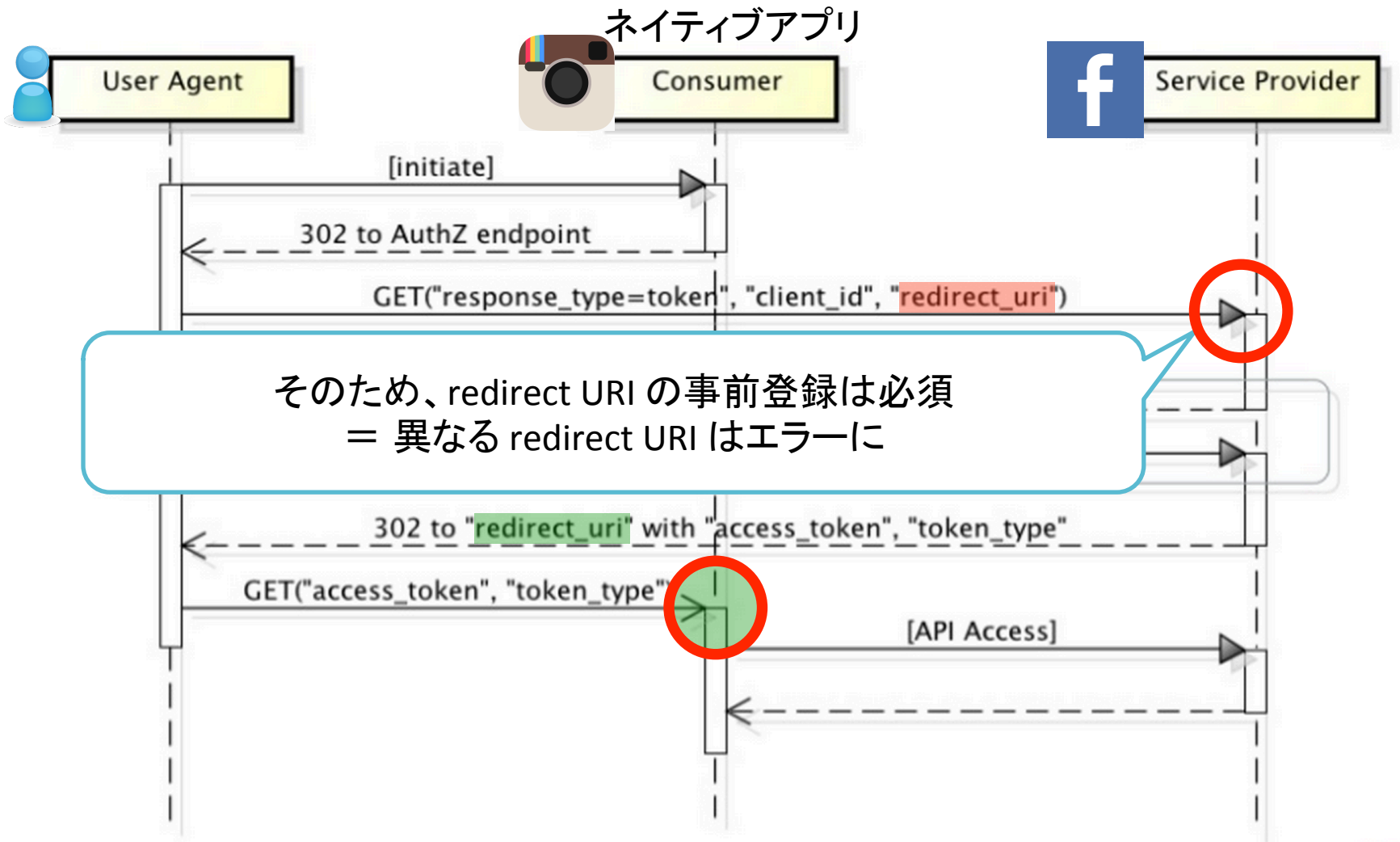
Implicit Grant 処理の流れ



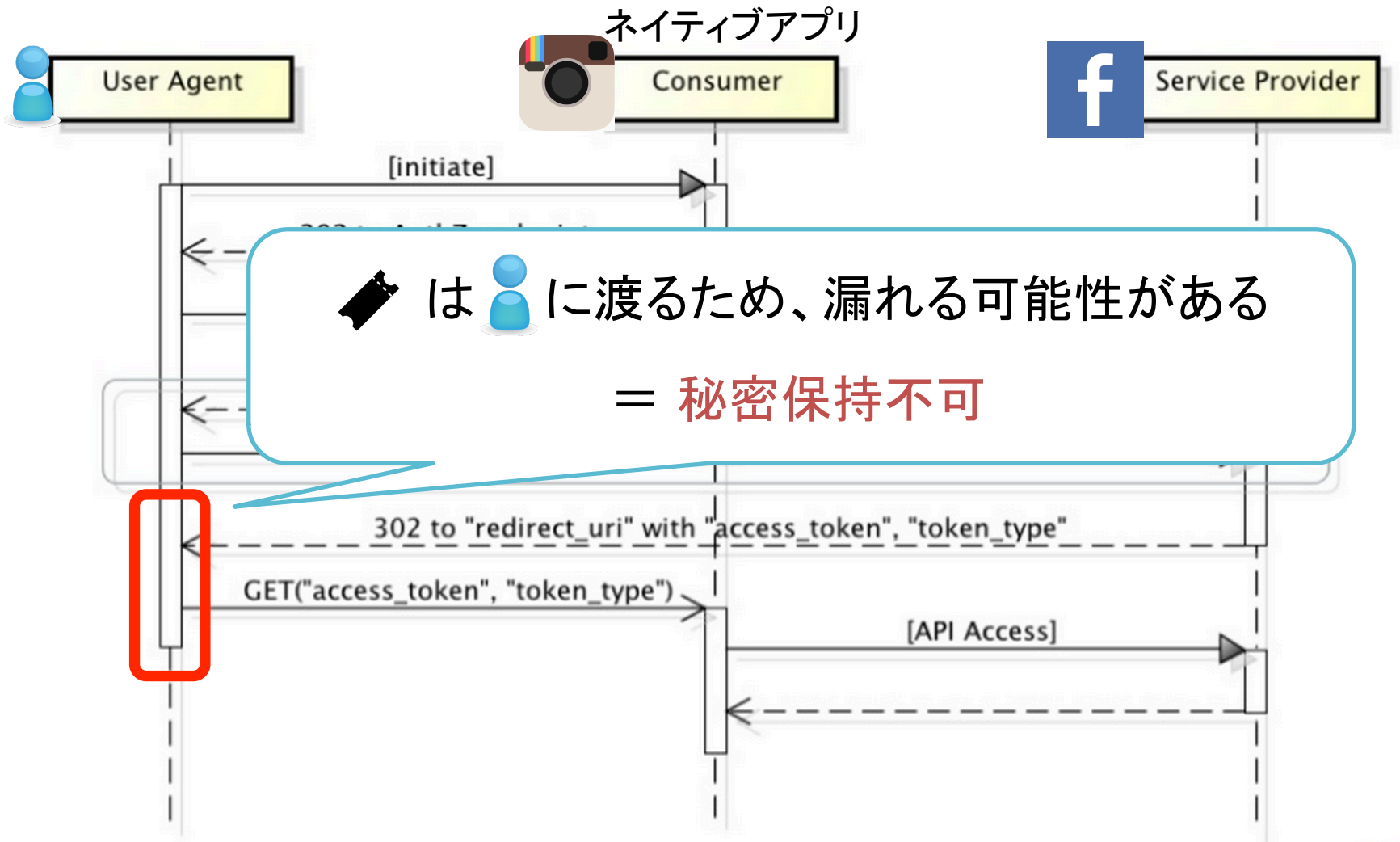
Implicit Grant 処理の流れ



Implicit Grant 処理の流れ



Implicit Grant 処理の流れ



話す内容

1. OAuth ってなに？
2. OAuth2.0 システム仕様
3. セキュリティ対策



Case1. CSRF

CSRF

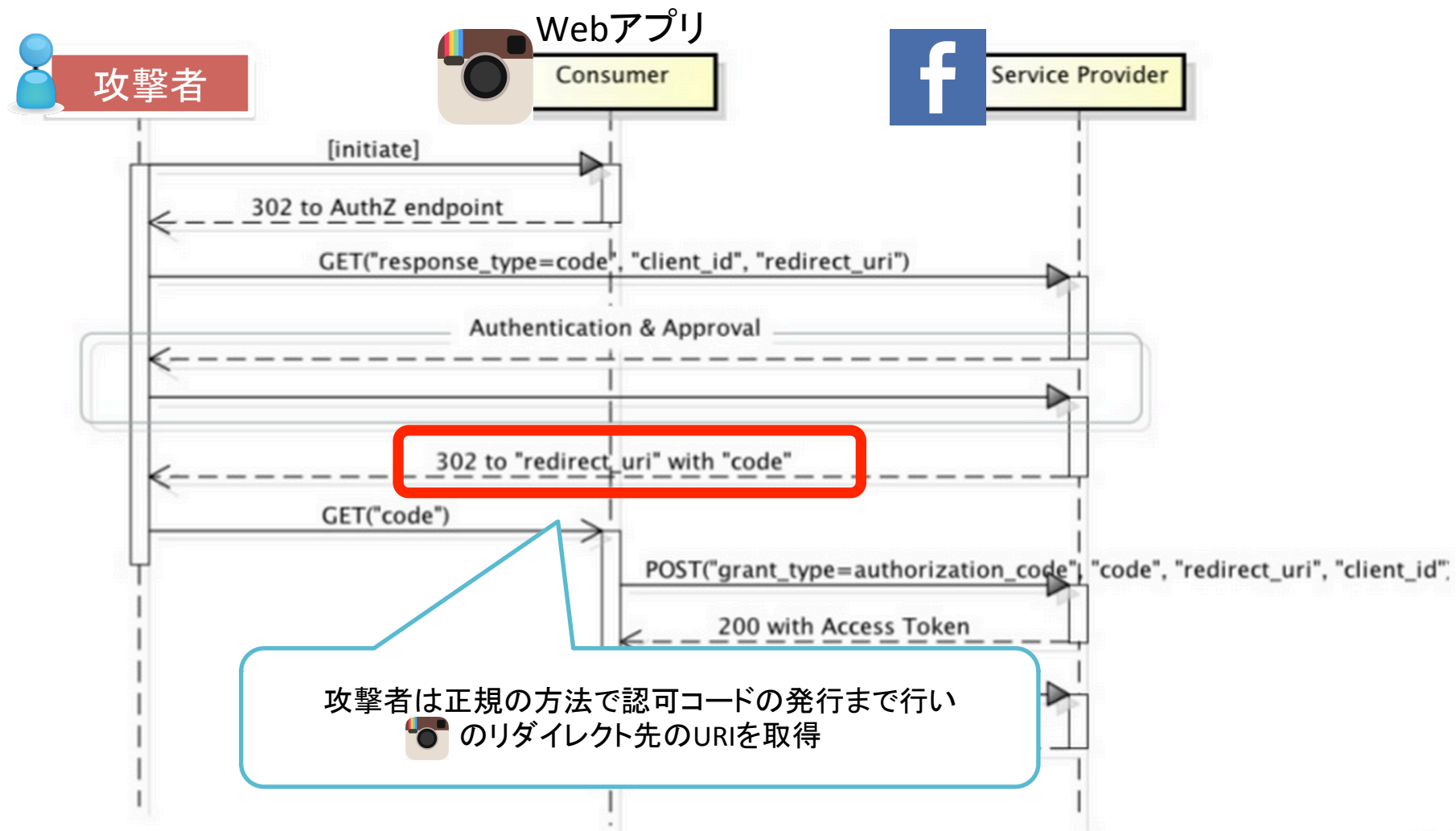
手法

- 1) 攻撃者は通常の手順でSPの認可まで行う
- 2) 認可後のリダイレクト先の URI を取得し、被害者にアクセスさせる

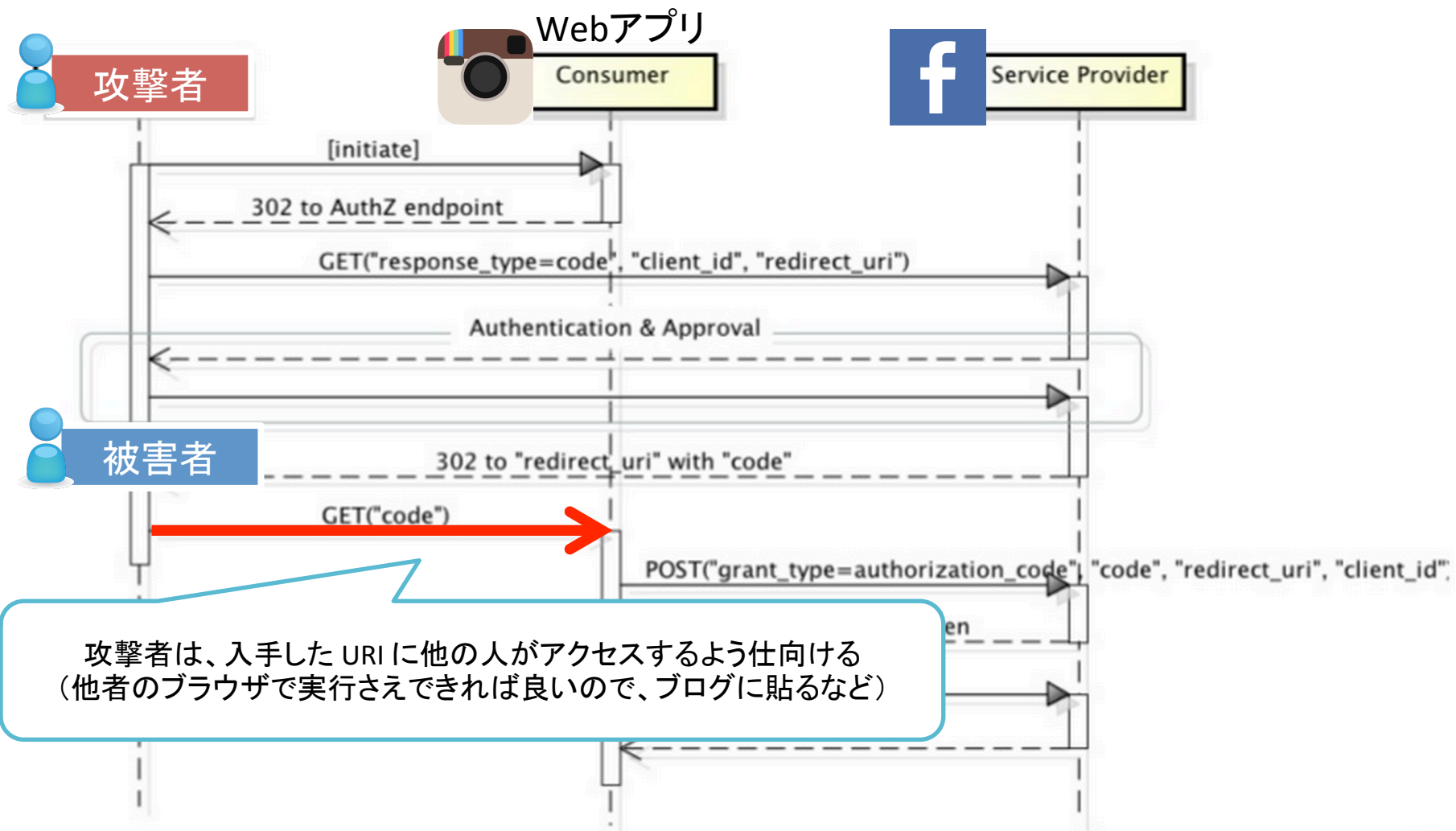
リスク

- 第3者のアカウントとしてログインさせられる
 - 個人情報やクレカ情報を登録したら、第3者に知られてしまう
- 第3者の  アカウントと連携
 - = 第3者がログイン可能
 - =  アカウントの乗っ取り

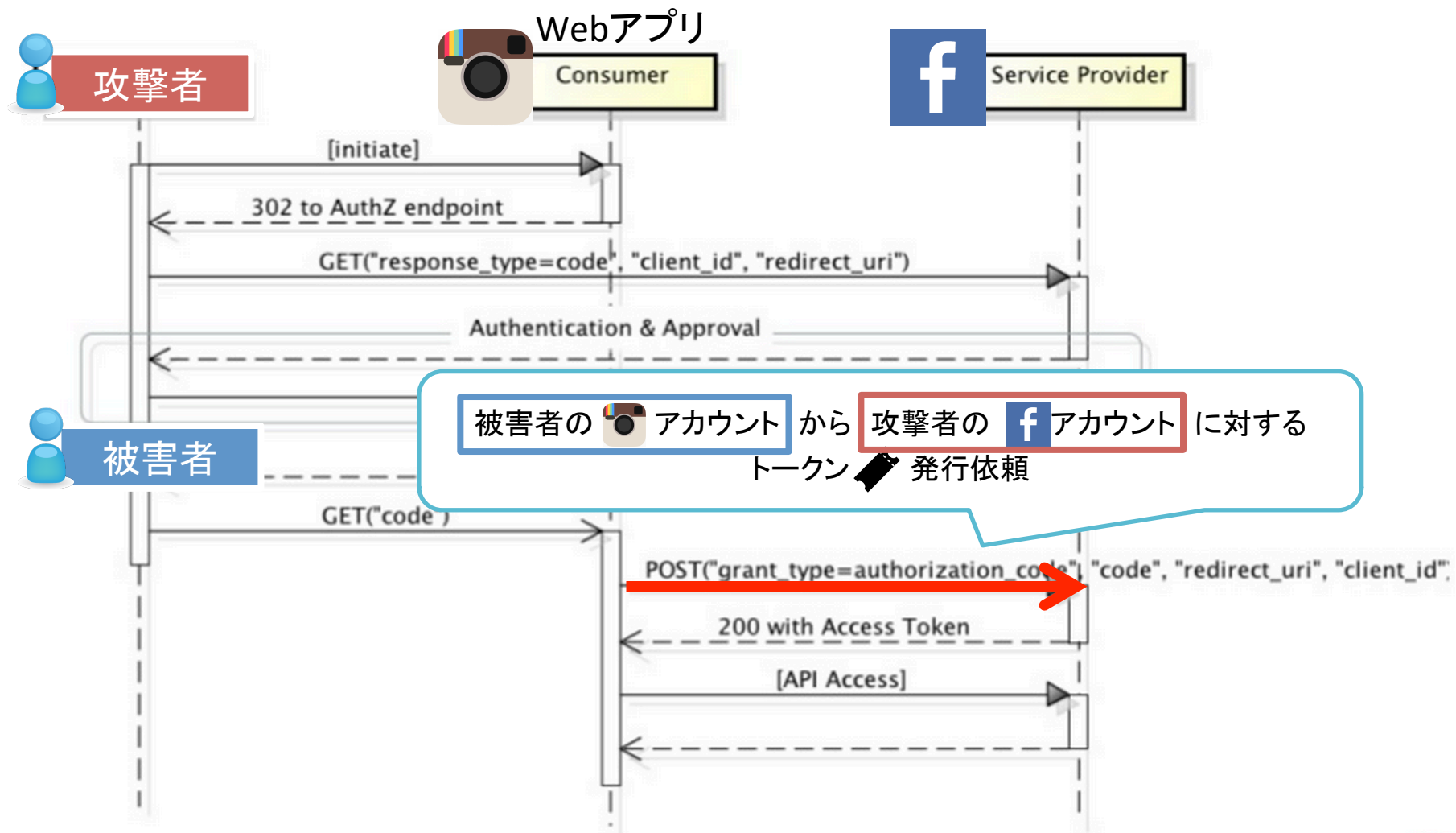
CSRF



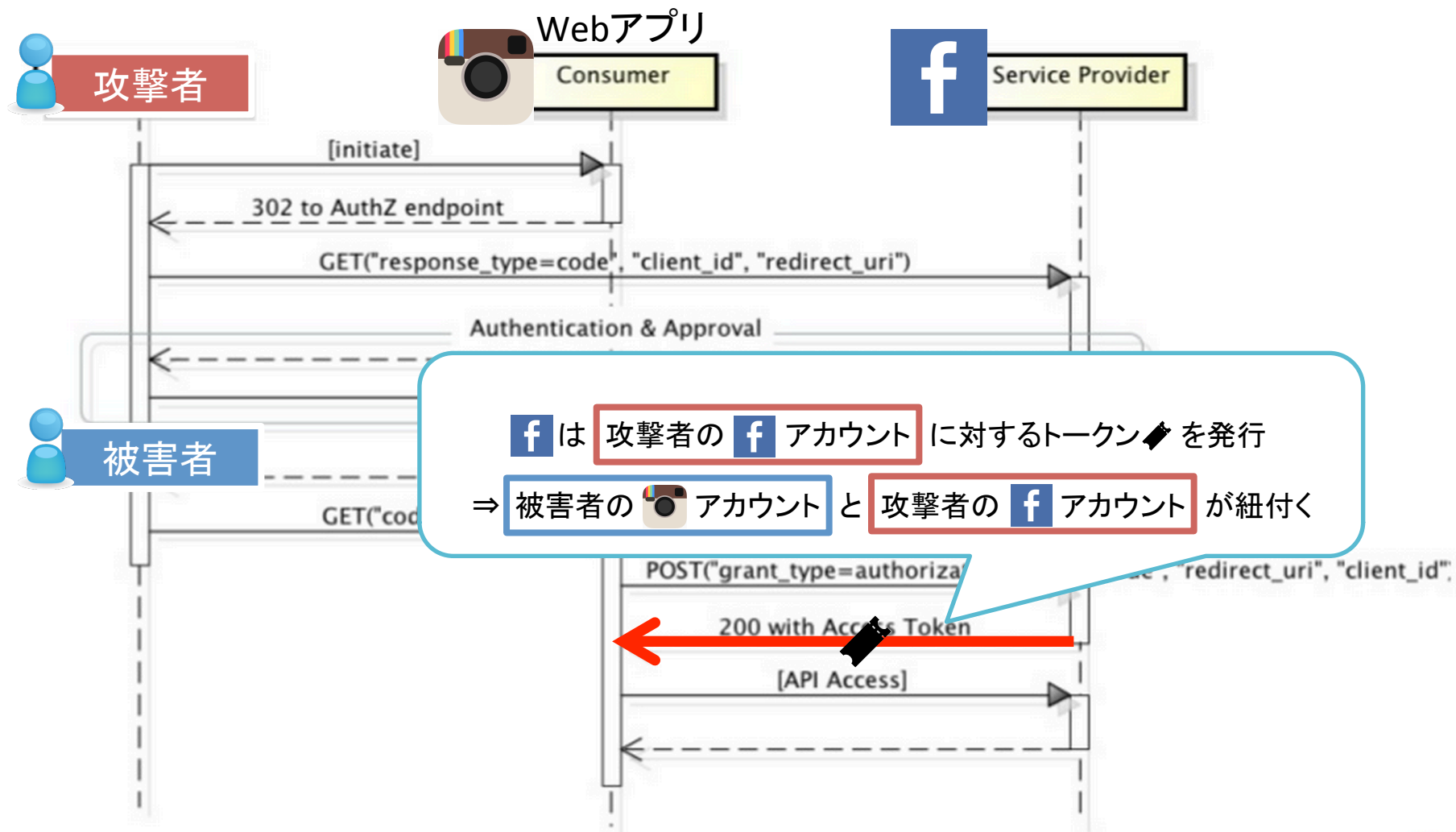
CSRF



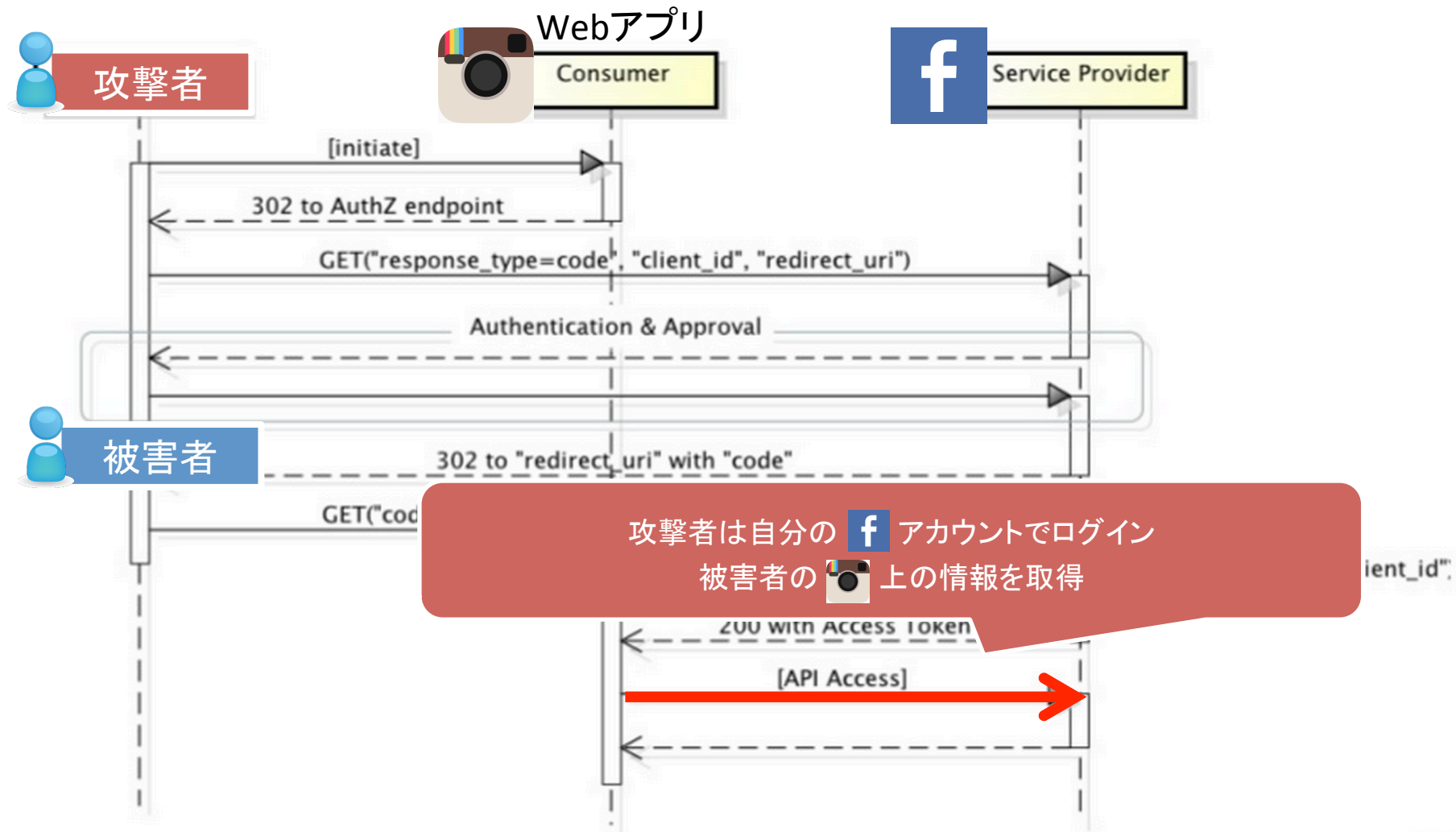
CSRF



CSRF



CSRF

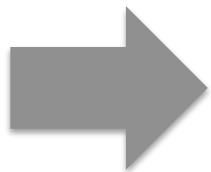


対策 (Consumer)

「ユーザー認可リクエスト」

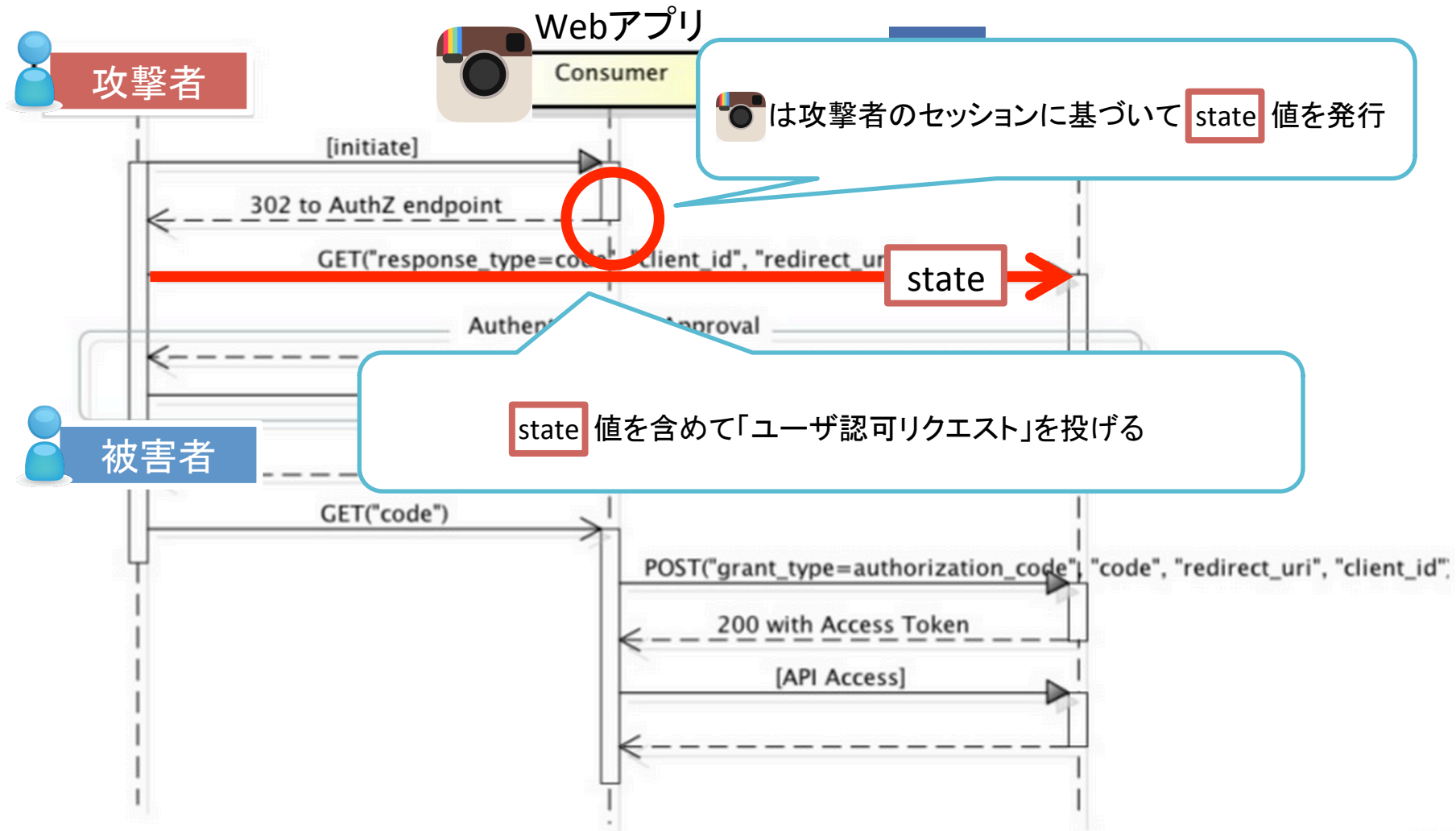
「アクセストークンリクエスト」

この2つが同じセッションかをチェック

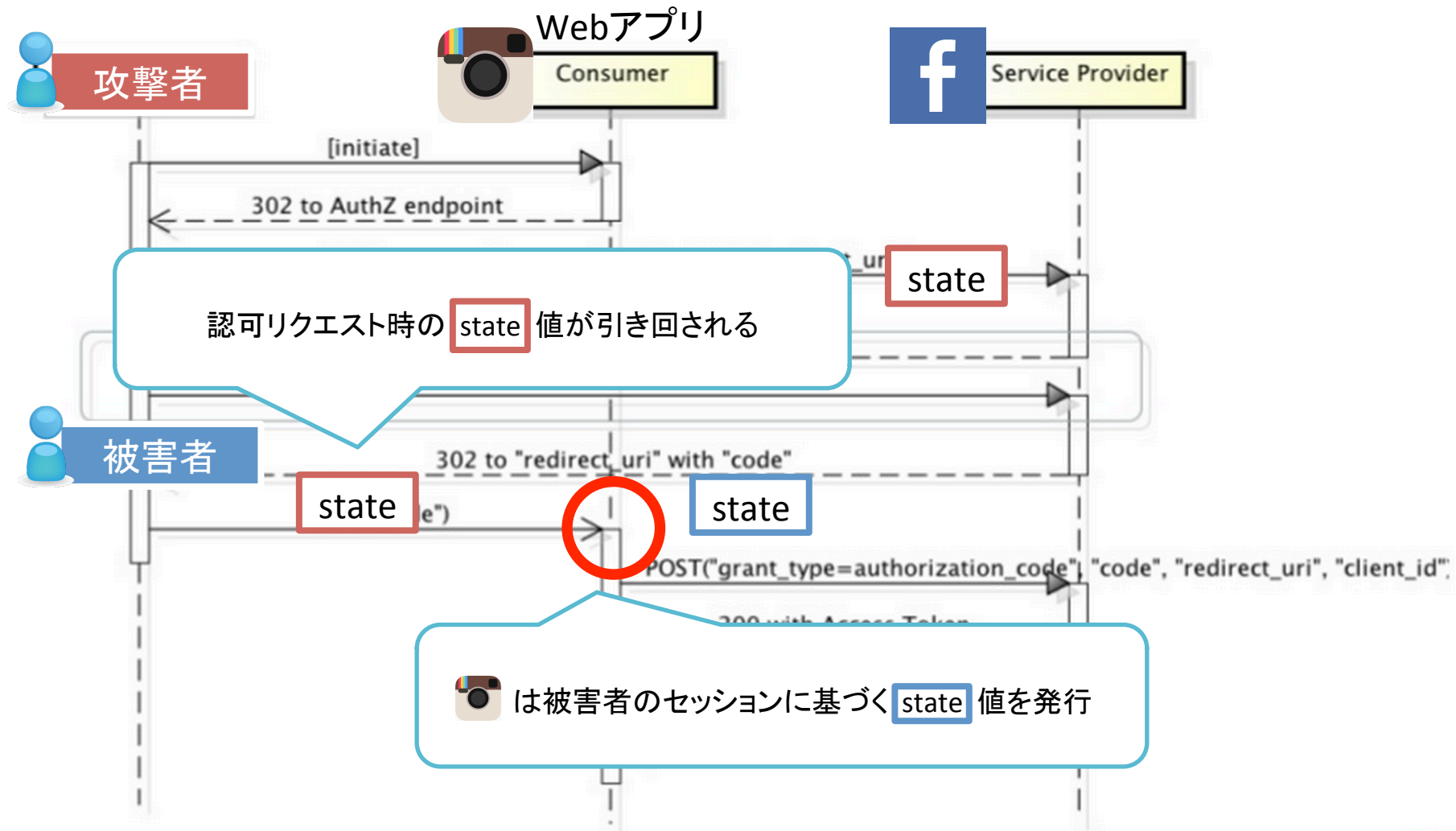


state パラメータを使う

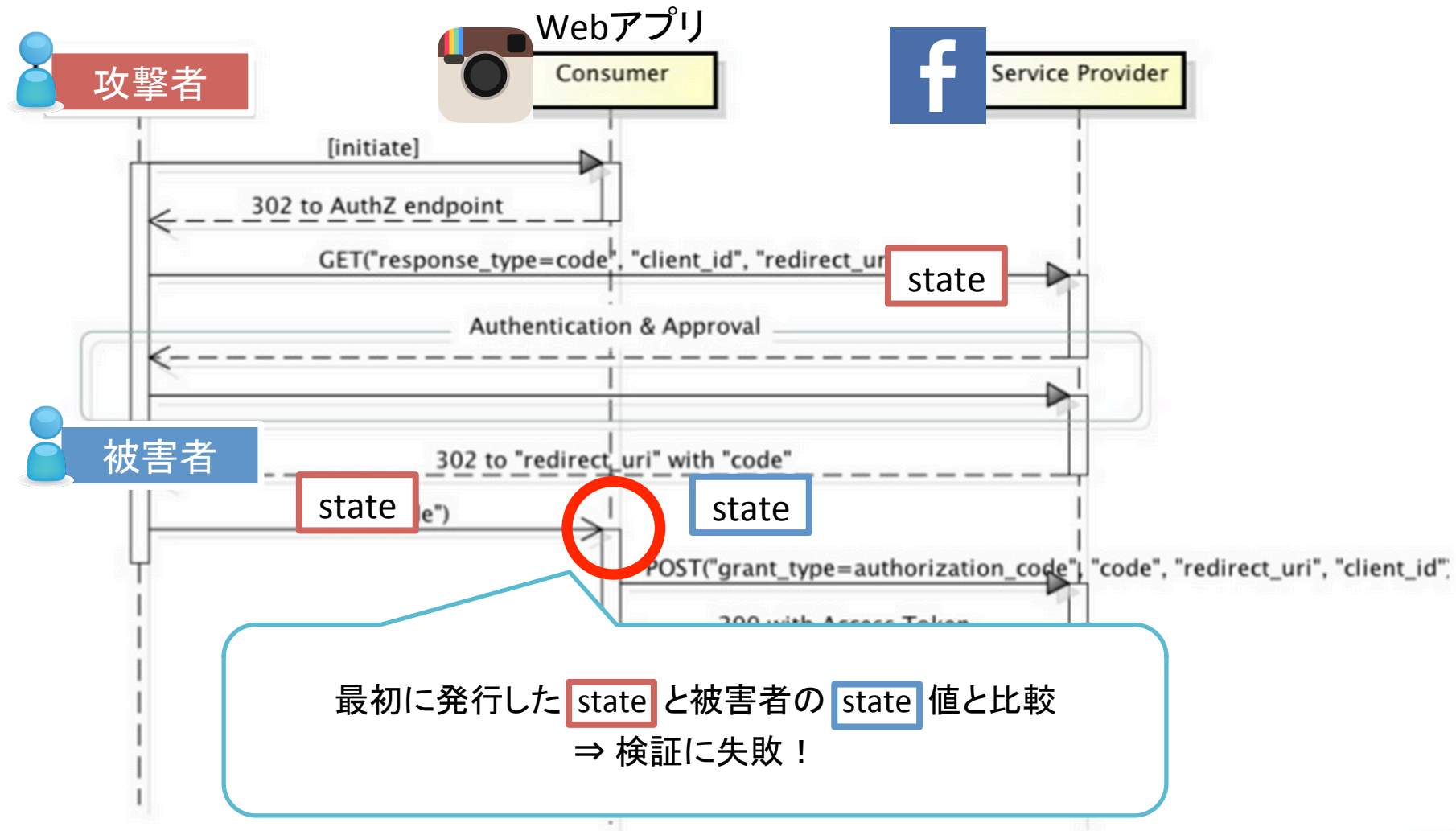
CSRF 対策



CSRF 対策



CSRF 対策



Case2. redirect URI の改ざん

redirect URI の改ざん

手法

「ユーザー認可リクエスト」と「アクセストークンリクエスト」で異なる redirect URI を指定

リスク

攻撃者にアクセストークンを抜き取られ、被害者の個人情報にアクセスされる、など

redirect URI の改ざん

手法

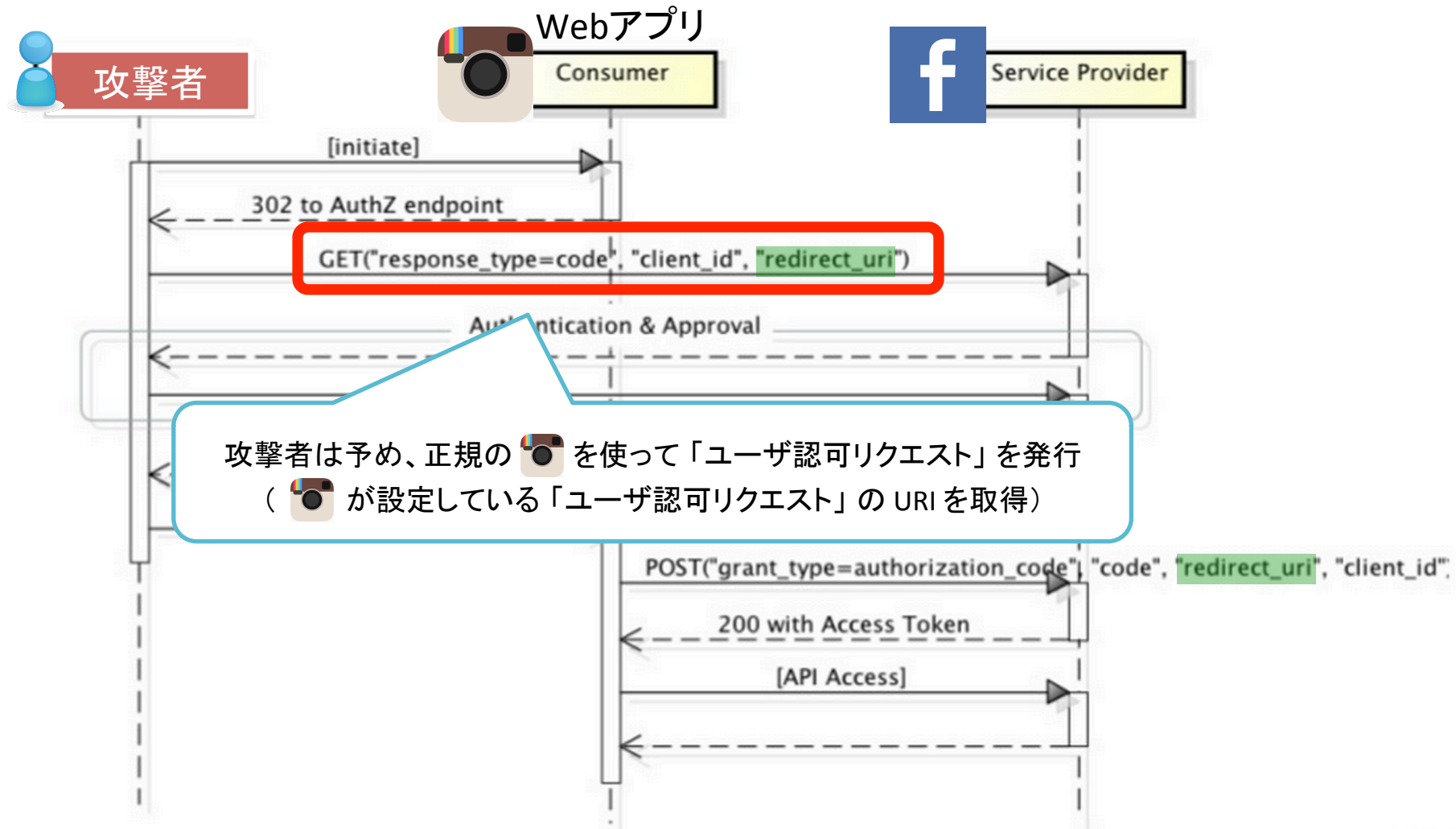
「ユーザー認可リクエスト」と「アクセストークンリクエスト」で異なる redirect URI を指定

条件

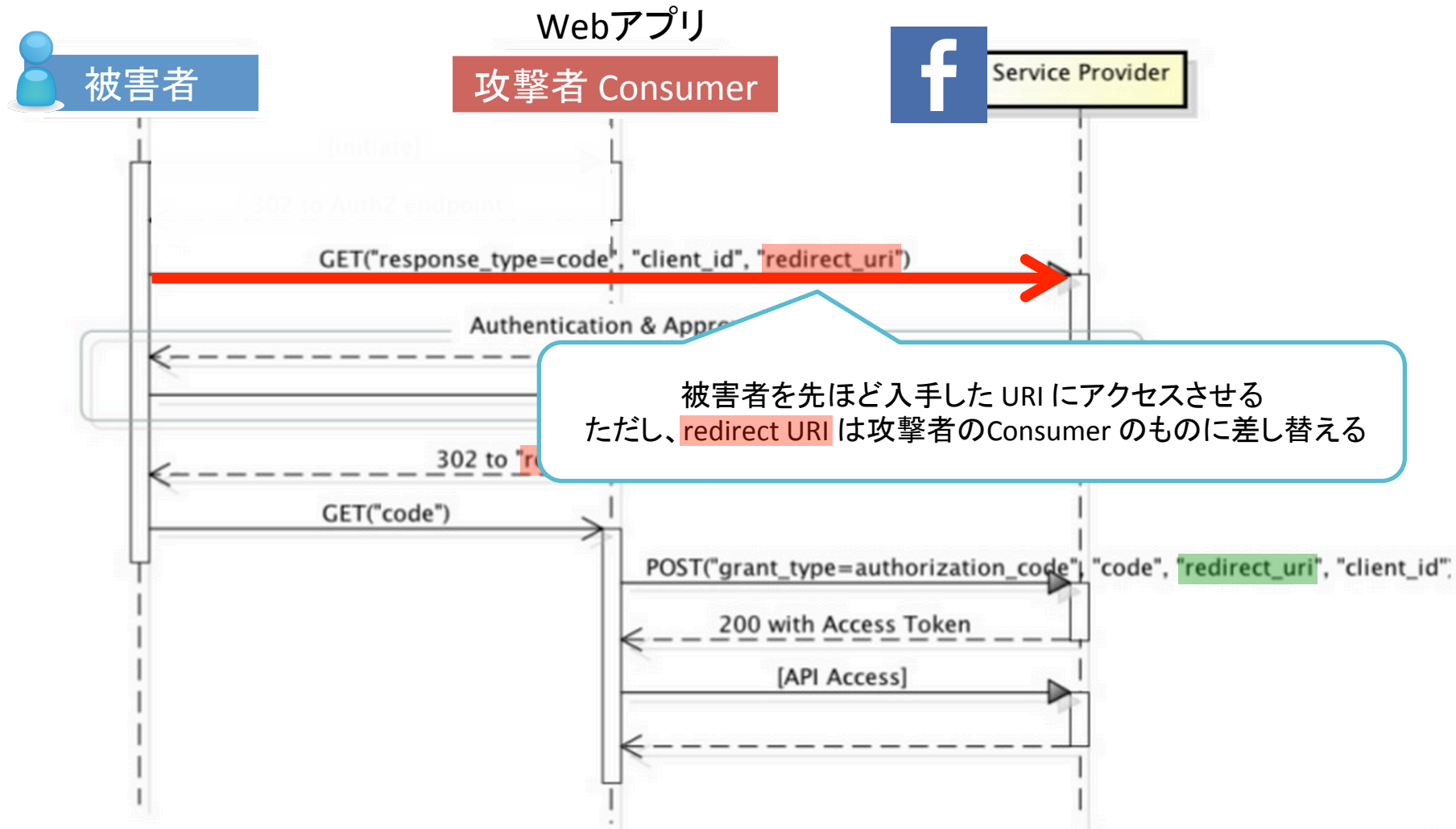
SP に redirect URI の事前登録をしない場合に発生

※Implicit Grant はSP への redirect URI の事前登録が必須なので、Authorization Code Grant に限られる

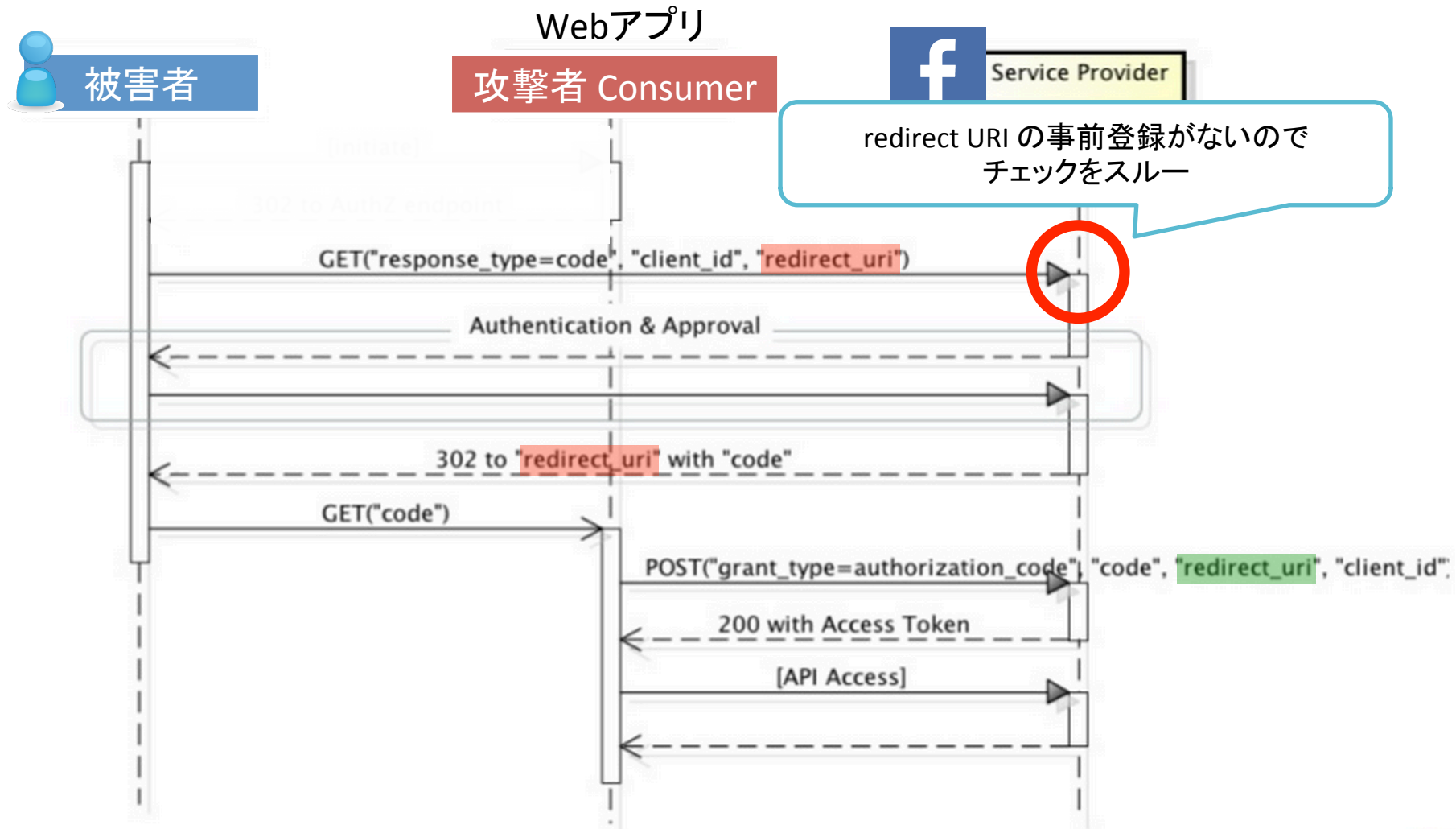
redirect URI の改ざん



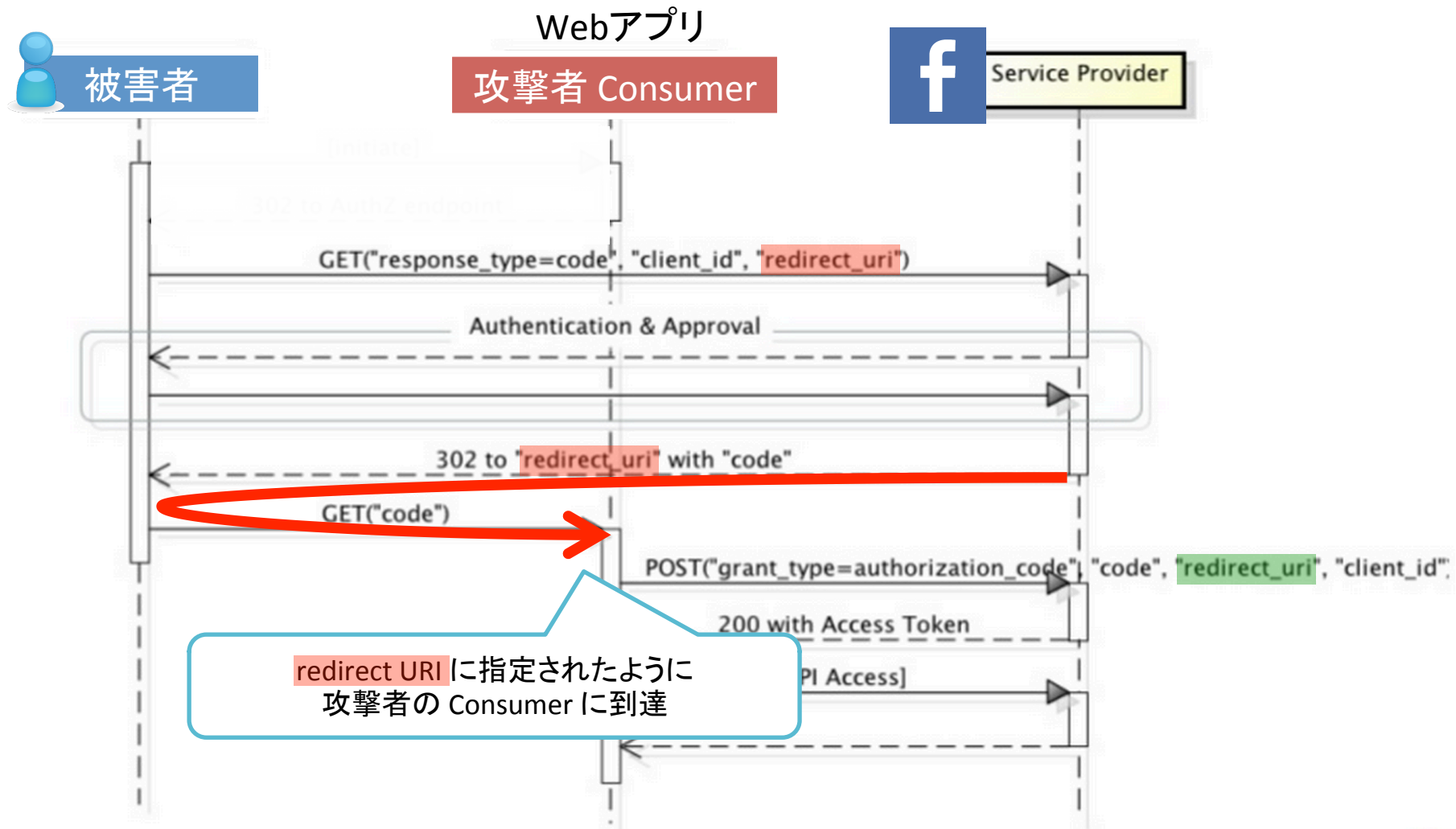
redirect URI の改ざん



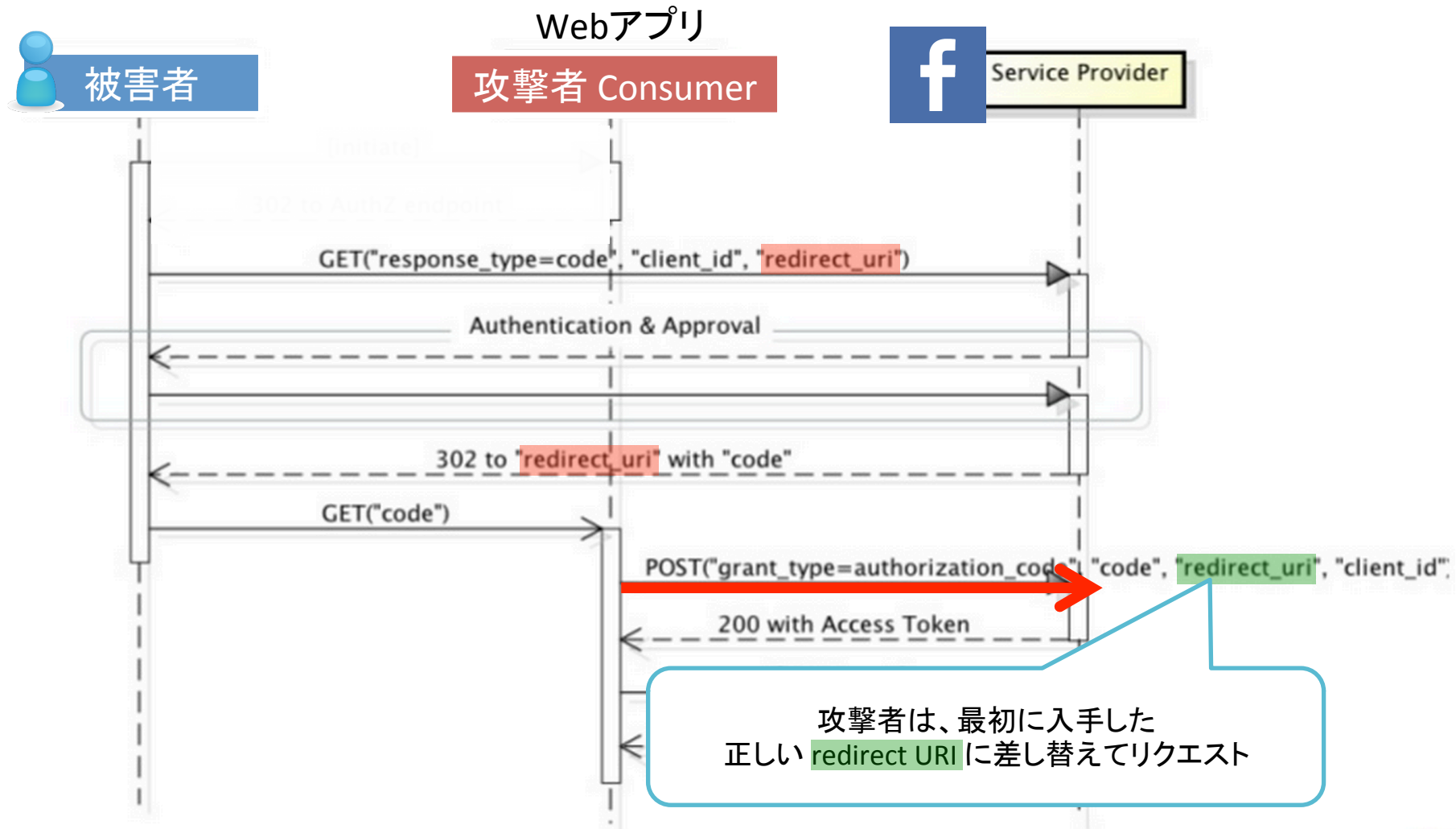
redirect URI の改ざん



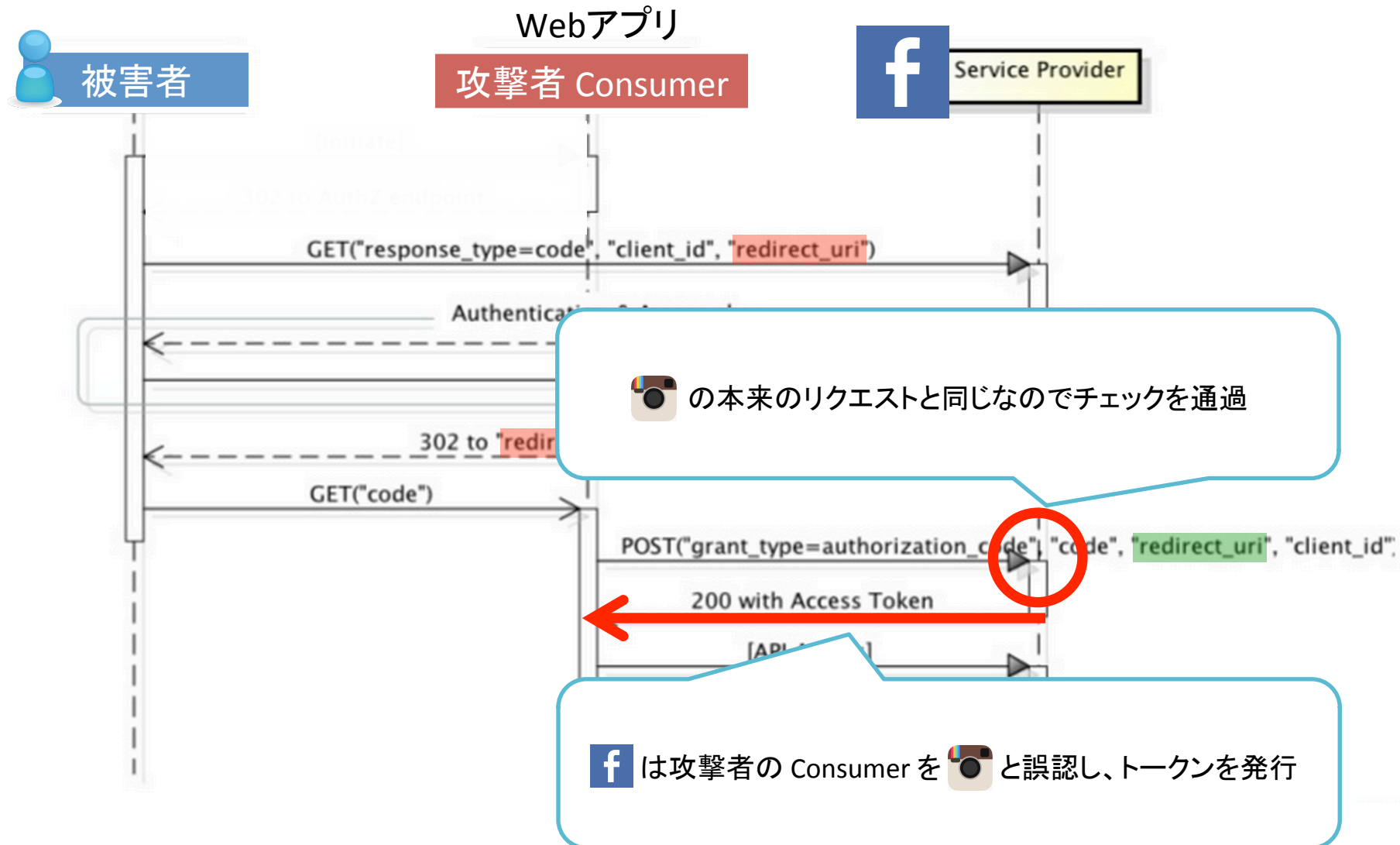
redirect URI の改ざん



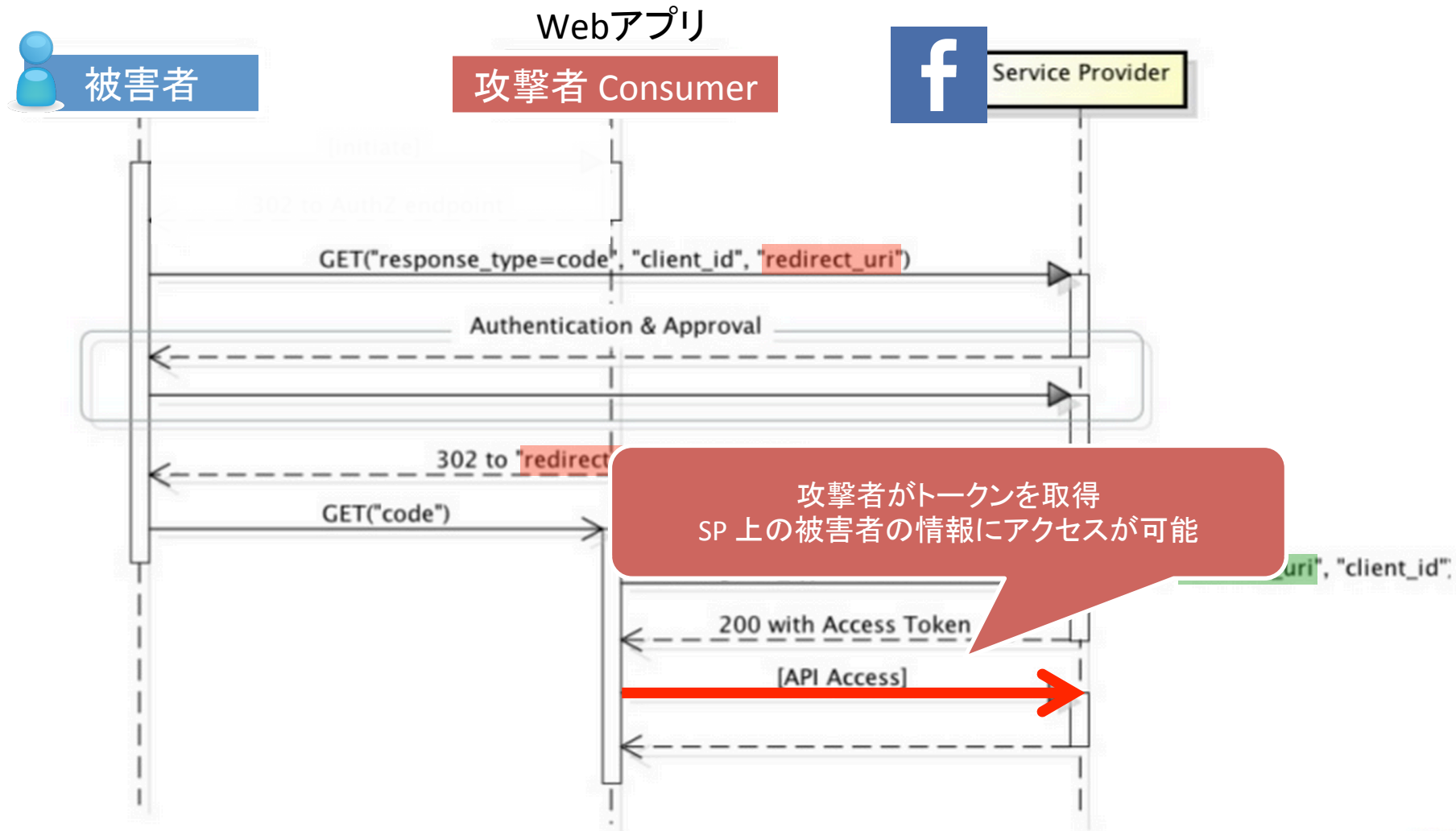
redirect URI の改ざん



redirect URI の改ざん



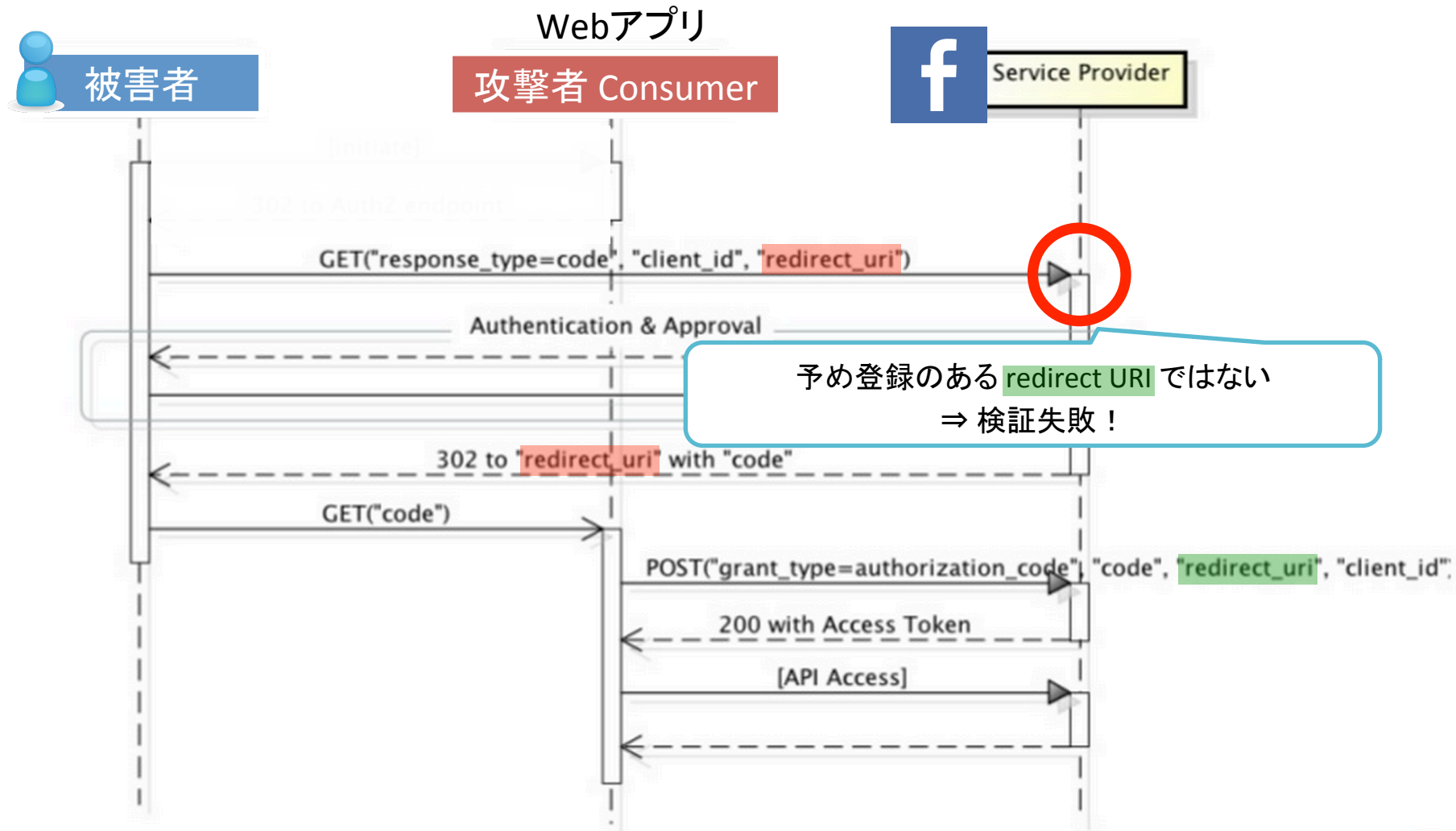
redirect URI の改ざん



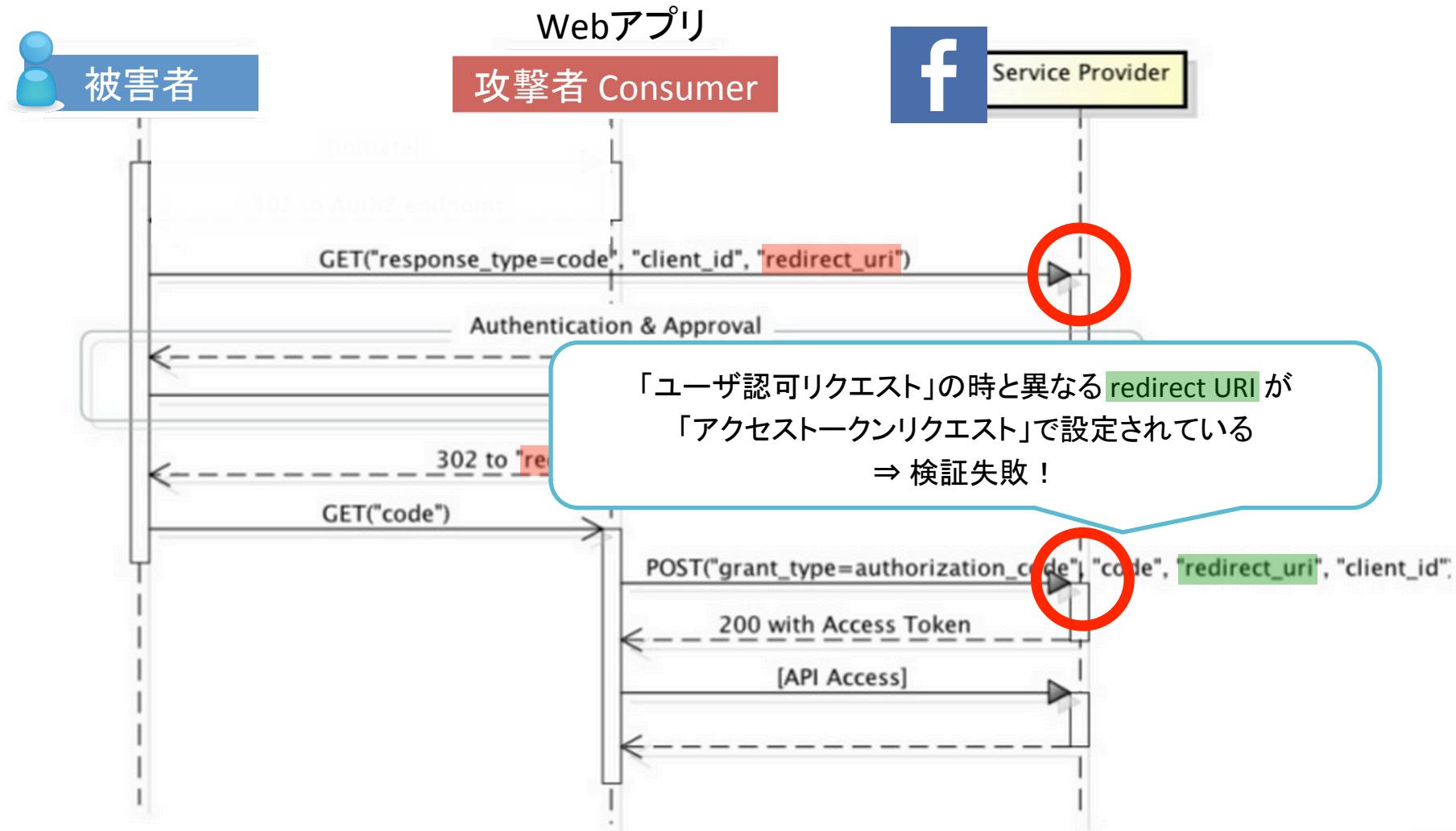
対策 (SP)

- 1) redirect URI の事前登録をしておき、登録のあるURIと同じかチェック
- 2) 「ユーザー認可リクエスト」と「アクセストークンリクエスト」で redirect URI が一緒かチェック

redirect URI の改ざん 対策



redirect URI の改ざん 対策



他にも・・・

攻撃例

- Consumer のなりすまし
- オープンリダイレクタ
- Implicit Grant におけるトークン置き換え

などなど・・・(説明は逐次追記します)

まとめ

まとめ

- OAuth とは認可のプロトコル
- トークンはあくまで「何ができるか」を証明するもの。「誰か」を証明するものではない
- トークンは第三者でも使用可能。漏れないように十分に配慮する
- セキュリティ上の配慮する点が多い。
Consumer としても SP としても対策を怠らないこと

参考

- <http://openid-foundation-japan.github.io/rfc6749.ja.html>
- <http://www.slideshare.net/charlier-shoe/oauth-20-29540466>
- <http://www.atmarkit.co.jp/ait/articles/1209/10/news105.html>
- <http://www.slideshare.net/ritou/idcon17-oauth2-csrfprotectionritou>
- <https://speakerdeck.com/ritou/openid-connectwoyong-iteidlian-xi-woshi-zhuang-surutokiniqui-wofu-kerukoto-number-yapcasia>
- http://developer.yahoo.co.jp/yconnect/client_app/