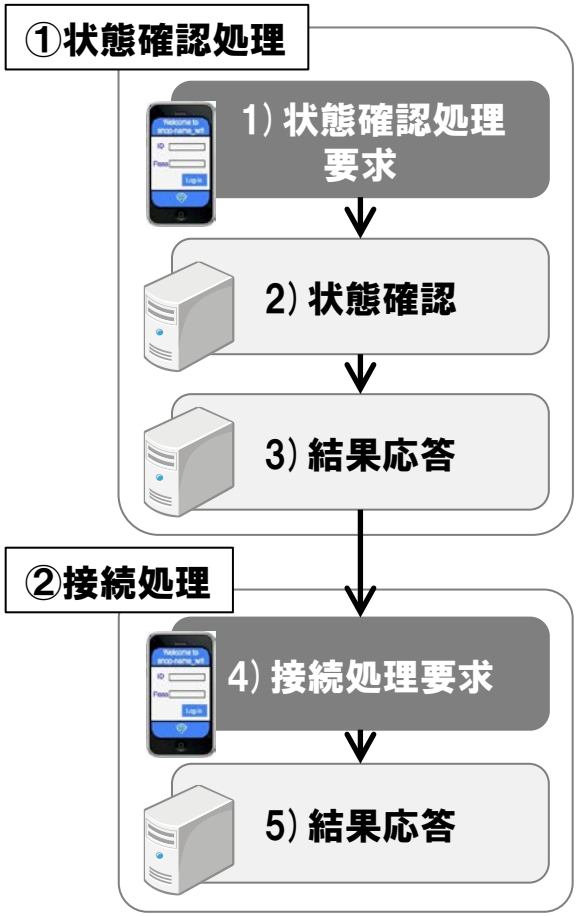
利用手続きの簡素化・一元化に係る実証実験
Web API インタフェース仕様書

2016年2月19日

目次

1. 仕様概要

- アプリからのWi-Fi接続は、下記2段階での認証処理を行うものとする。
  - ① 状態確認処理　　アプリ自体が認可されたものかどうかを判定
  - ② 接続処理　　アプリが認可されたものであった場合に、個々のユーザ識別を伴う認証を行う



1) 接続アプリは、認可アプリ毎に事前に払い出されたID/PWを含んだ「状態確認処理リクエスト（HTTP）」を行う。

2) 状態確認処理リクエストを受けたネットワークは、「許可アプリIDチェック」及び「インターネット接続済み又は未認証状態のチェック」を行う。

3) ネットワークは、「インターネット接続済み」又は「未認証状態」かを返却する。

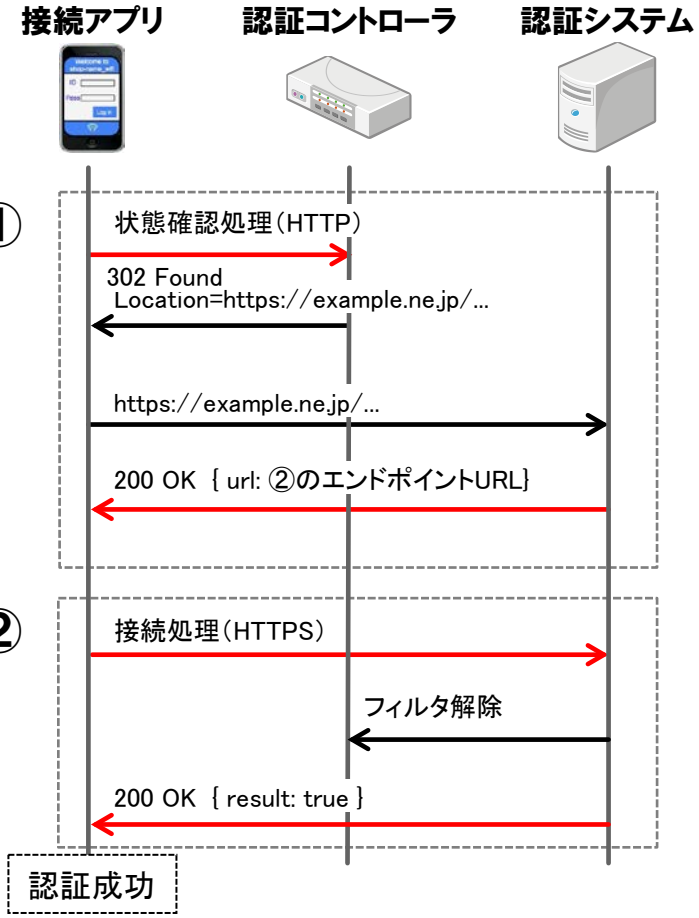
4) 接続アプリは、「未認証状態」だった場合のみ、接続処理のリクエストを行う　（HTTPS）

5) ネットワークは、認証及びフィルタ開放処理を行い、結果を接続アプリ　へ返却する。

2. 接続シーケンス(1/3)

- 認可アプリでのみ利用可能とし、アプリ毎に払い出されたID/PWによる認証を必須とする。
- 認証時に使用されるIDを、APPIDおよびAPIバージョン情報としてネットワークは使用する。

＜認証成功時＞



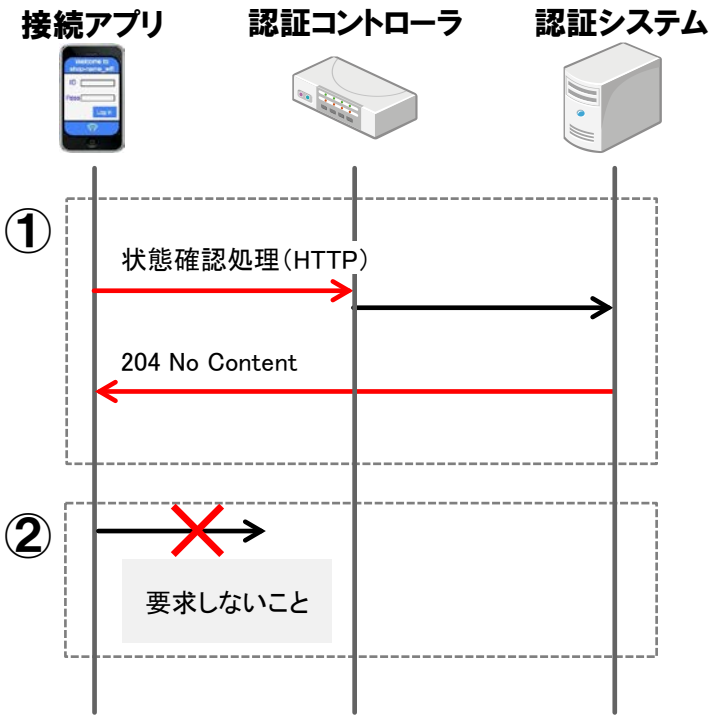
【①状態確認処理】
接続アプリは認証前に状態確認処理のリクエストを行い、ネットワークは認可アプリチェック、インターネット接続済等の確認を実施し、状態確認処理OK後にレスポンスを行う。接続アプリは、レスポンスに含まれるURLを認証時のエンドポイントとして使用する。

【②接続処理】
接続アプリは、エンドポイントURLへ接続処理のリクエストを行う。ネットワークは認証処理＆フィルタ解除処理を実施し、レスポンスOKを返却。
リクエスト/レスポンスのPayloadはJSONとし、将来拡張のため接続アプリ/ネットワークともにJSON Object内の不要項目は読飛ばすものとする。

2. 接続シーケンス(2/3)

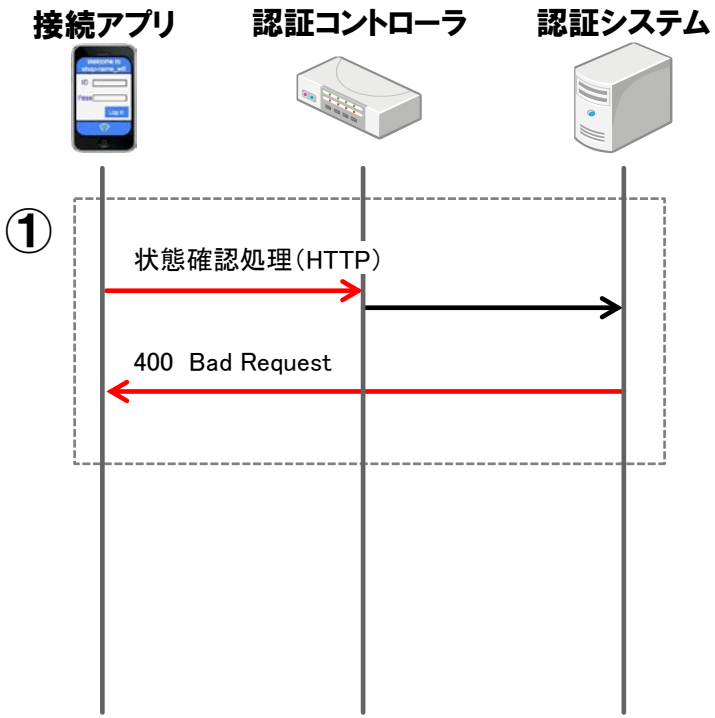
- 「①接続処理」において、既にインターネット接続済みの場合、ネットワークは204 (No Content) で応答する。

＜インターネット認証済＞



- 「①状態確認処理」が失敗した場合、ネットワークは400 (Bad Request) で応答する。接続アプリは、40x/50xを通信エラーとして処理する。

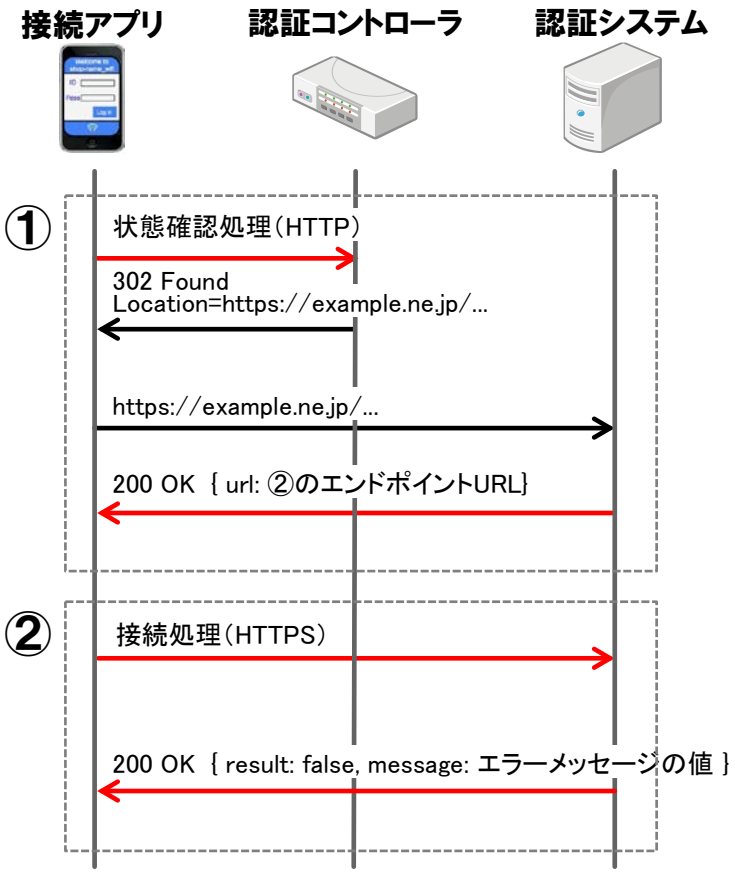
＜状態確認処理失敗＞



2. 接続シーケンス(3/3)

- 「②状態確認処理」が失敗した場合。ネットワークは200（OK）で応答し、エラーメッセージの値を返す。

＜認証失敗時＞



3. API仕様　①状態確認処理(1/2)

- 各事業者は予め、接続を許可するアプリのアプリIDとシークレットキーを、認証システム側で管理しておくこと。
  一方、アプリ開発者側では、そのアプリIDとシークレットキーを埋め込むこと。

- 状態確認処理のリクエストはHTTPとし、各事業者別に指定されたURLに対してリクエストを送信すること。

リクエストURL	http://<各事業者指定ホスト>/<各事業者指定パス>?<パラメータ群>
HTTPメソッド	GET

- リクエスト生成時のパラメータ群は以下のように生成すること。

appid	事前管理しているアプリの固定ID
nonce	UNIX Epochからの経過秒数(UTC)
token	シークレットキーを使って生成したハッシュ値
uiid	接続アプリ側で管理しているユーザアイデンティティ 事業者側は、認証ユーザとして扱わず、トレーサビリティとしてログ保持するのみ

＜状態確認処理リクエスト例＞

```
GET http://example.com/chk?appid=abcde
&nonce=1449728757&token=d13f1ce5e985614b7af253daa5&uiid=0001-abcde123
HTTP/1.1
```

3. API仕様　①状態確認処理(2/2)

- 接続アプリは、「状態確認処理」時にHTTP 302 (Found) によるリダイレクト処理および
  HTTPSへの切り替えに追従すること。
  HTTP 302はHTTP Clientライブラリ側でリダイレクト処理されるため、
  HTTP 200、204をコール側で正常系と して処理し、それ以外を通信エラーとして処理する。

- ネットワークは接続アプリ認証処理を行い、200 (OK) で以下のJSONで応答すること。
  urlは接続処理時のエンドポイントとする。

```
200 OK
Content-Type: application/json
Transfer-Encoding: Chunked
 {
    “url”: “https://example.ne.jp/auth?aaa=xxx....”
}
```

3. API仕様　②接続処理API(1/2)

- **接続処理のリクエストはHTTPSとし、**
  ①状態確認処理 で取得したURLに対してリクエストを送信すること。

リクエストURL	状態確認時に返却されたURL
HTTPメソッド	POST

- **リクエスト生成時のパラメータ群は、各事業者別で指定すること**

＊＊＊＊＊	＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
＊＊＊＊＊	＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
＊＊＊＊＊	＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊

各事業者別でパラメータ指定

＜接続処理リクエスト例＞

```
POST https://example.ne.jp/auth?aaa=xxx....  HTTP/1.1
Content-Type: application/json
Content-Length: xxx
 {
    各種パラメータ
}
```

3. API仕様　②接続処理API(2/2)

- ネットワークは接続処理を行い、200 (OK) で以下のJSONで応答する。
  resultで成否を判定し、接続完了時のmessageは常にnullとする。

```
200 OK
Content-Type: application/json
Transfer-Encoding: Chunked
 {
    "result": true,
    "message": null
}
```

- 接続処理が失敗した場合は、200 (OK) で以下のJSONで応答する。
  resultで成否を判定し、 messageは下表の通りとする。

```
200 OK Content-Type: application/json
Transfer-Encoding: Chunked
 {
    "result": false,
    "message": “S_0008“
}
```

値	内容	
A_0005	LOGIN_LIMIT_EXCEEDED	日次制限回数超過
S_0008	OUT_OF_SERVICE_HOURS	利用可能時間外
S_0009	OUT_OF_SERVICE_AREA	利用可能エリア外
B_0006	SYSTEM_ERROR	システムエラー
N_0002	EXPIRED_ENTRYPOINT	セッション情報の不整合発生