JNSA 2017年度 電子署名WG 夏合宿

- 現在のCSC仕様を読み解く -

LangEdge,Inc.
有限会社 ラング・エッジ

宮地（miyachi@langedge.jp）

２０１７年９月１日

はじめに「現在のCSC仕様を読み解く」

CSC（Cloud Signature Consortium）**仕様書**
CSC_0.1.7.9_PR.pdf「Architectures, Protocols and API Specifications for Remote Signature applications」（全60ページ）をチェックした結果をまとめる。現在のバージョンは以下、

Public pre-release version 0.1.7.9-PR (2017-02-14)

ドキュメント取得：
http://www.cloudsignatureconsortium.org/specifications/
※ ドキュメント取得には所属・氏名・メールアドレス・CSC参加希望 の登録が必要。

つまり現在はプレリリース版であり、今後改定される可能性がある点に注意が必要。おそらく拡張の方向で改定される。

- 関連した資料のページ番号を右下に記載する。

関連：CSC_0.1.7.9_PR ページ 1

CSC仕様書 目次

1 Scope [スコープ]
2 Requirements Language [要求言語]
3 References [参照]
4 Terms and definitions [用語定義]
5 Conventions [表記]
6 Architectures and use cases [構造とユースケース]
7 Entities and components of a Remote Signature solution [リモート署名要素]
8 Introduction to the Remote Service Protocols API [概要]
9 Authentication and authorization [認証/認可]
9.1 Service authorization and authentication [サービス認証/認可]
9.2 Credential authorization [クレデンシャル認可]
9.3 OAuth 2.0 Authorization [OAuth 2.0 認証]
10 Creating a Remote Signature [リモート署名生成]
10.1 Multi-signature Transactions [複数署名トランザクション]
11 Error handling [エラー処理]
12 The Remote Service APIs [リモートサービスAPI仕様]
13 Interaction among elements and components [シーケンス図]

関連：CSC_0.1.7.9_PR ページ 2-3

CSC標準スコープ外

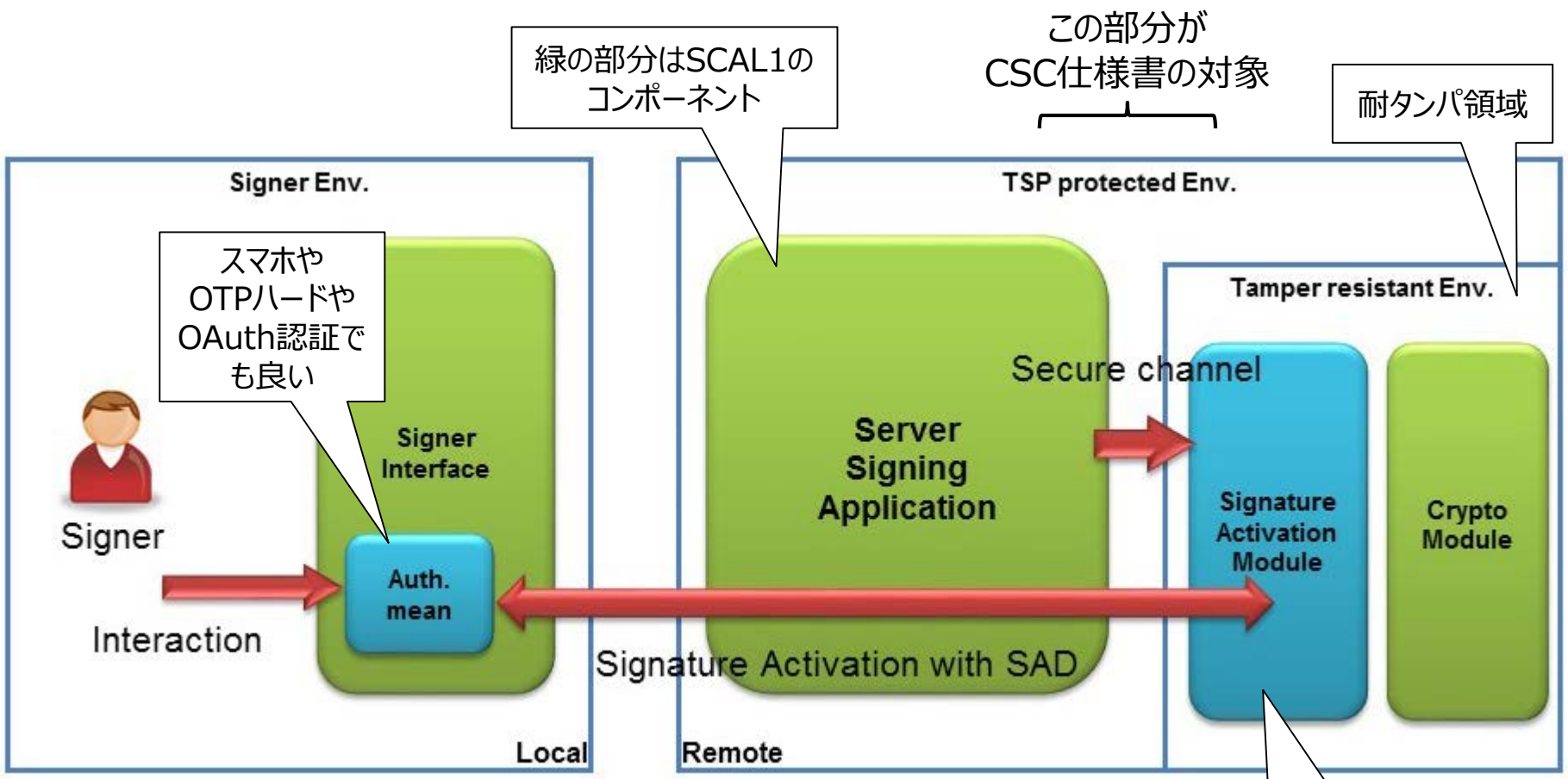
- **サービスプロバイダーのポリシー要件**
  ⇒ ETSIが対象とする標準化の分野。
- **署名および証明書フォーマット**
  ⇒ ETSIで指定された規格の使用を推奨。
- **署名鍵保持ハードウェアのセキュリティ評価/要件**
  ⇒ CENと米国のFIPSによって標準化済み。
- **サービスプロバイダのシステム内の内部仕様**
  ⇒ 理由は書かれていないが、APIとプロトコルが合っていれば良い？
- **署名の検証**
  ⇒ CSCの**今後の仕様書**で取り上げられる予定。

■ **個人的装置に署名鍵があるケースは現在対象外**
⇒ CSCの**今後の仕様書**で取り上げられる予定。

□ **署名鍵の生成や証明書の登録は対象外**
⇒ CSCの対象は登録済みのところから。今後もCSCのスコープ範囲外か。

関連：CSC_0.1.7.9_PR ページ 6

アーキテクチャー図１

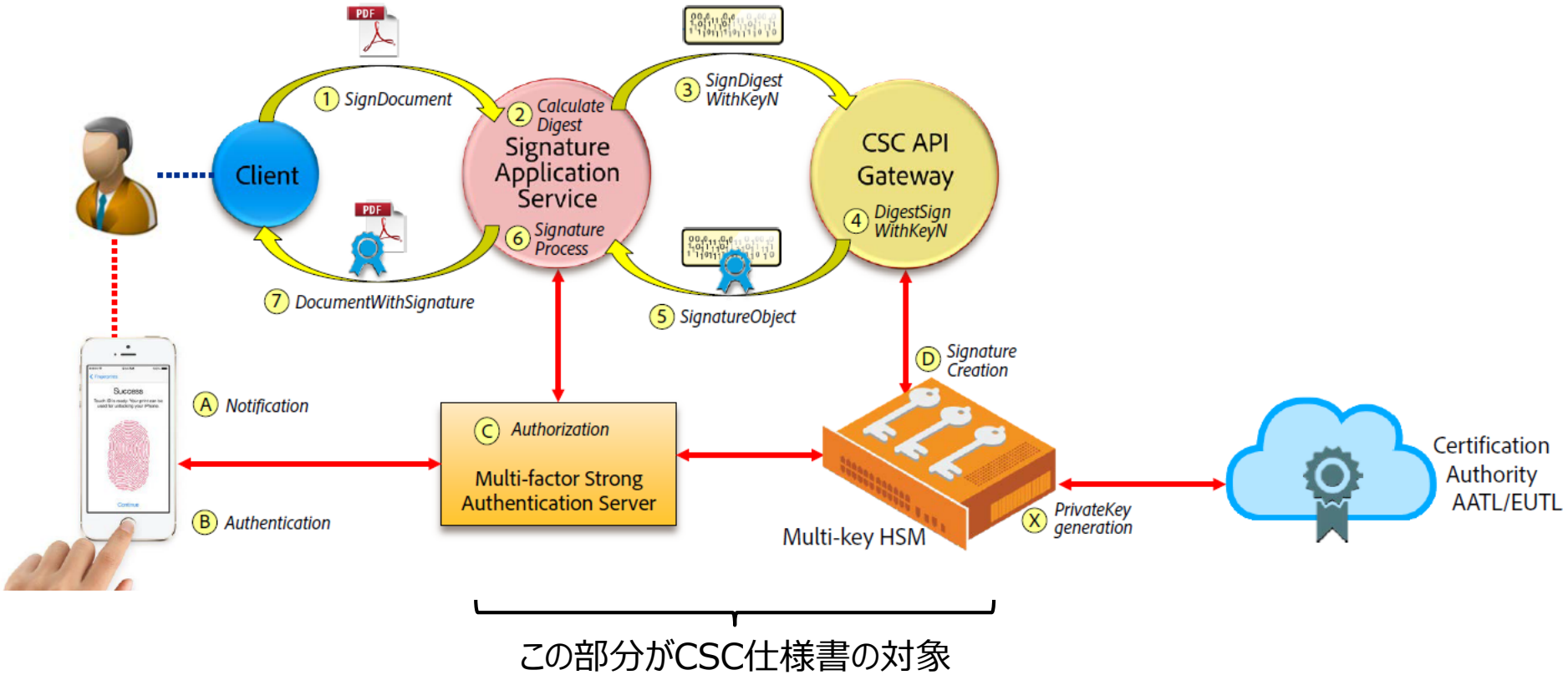


ETSI/CENのリモート署名で良く使われている図そのまま。
クレデンシャル認可SCAL2（単独制御保証レベル2）の図。
これだけ見せられても良く分からない。

関連：CSC_0.1.7.9_PR ページ 9

アーキテクチャー図３（CSCプレゼン資料より）

このアーキテクチャー図が一番まとも！
この図をCSC仕様書に入れるべき。



関連：Cloud_Signature_Consortium.pdf ページ 6

略語 – 赤字は構成要素

CAS: Credential Authorization Service ※

HSM: Hardware Security Module

RSCD: Remote Signature Creation Device

RSS: Remote Signing Service ※

RSSP: Remote Signing Service Provider

SCAL: Sole Control Assurance Level

SCSP: Signature Creation Service Provider ※

SA: Signature Application ※

SAD: Signature Activation Data

SAM: Signature Activation Module

SCA: Signature Creation Application ※

SSA: Server Signing Application ※

※ 本文中では使われていない…

関連：CSC_0.1.7.9_PR ページ 8,10

CSCのAPI基本仕様

基本フォーマット：REST+JSON

REST ⇒ HTTP POST（基本GETは使わない）

JSON ⇒ request body and response

ベースURI：以下の形式

https://service.domain.org/csc/v0

通信保護：TLS 1.2 / TLS 1.1

※ TLS 1.0は未サポート

リモートサービス情報：infoエンドポイント

ex) https://service.domain.org/csc/v0/info

関連：CSC_0.1.7.9_PR ページ 11-12

認証(Authentication)と認可(Authorization)

2種類の認証/認可：

a. サービスの認証/認可（署名サービス独自利用）

b. クレデンシャル認可（SAM利用秘密鍵利用許可）

サービスの認証/認可方式：

- **アクセストークンによる認可方式をサポート**
- 他の様々な認証/認可もサポート可能
- プライベートLANやVPNを使っても良い

サービスの認証に使える2種類のCSC認証API：

- ✓ **auth/login ⇒ Basic認証/Digest認証**
- ✓ **oauth2/token ⇒ OAuth 2.0 トークン取得**

※ OAuth 2.0 認証/認可のAPI（oauth2/authorize）はIdPを指定

関連：CSC_0.1.7.9_PR ページ 12-15

クレデンシャル認可（認証）

現在以下の3種類の認可方式をサポート：

- **Implicit authorization**（暗黙的認可）
  - ✓ 署名アプリケーションを**経由せず**直接SAMが利用者と認可プロセスを管理
  - ✓ SCAL1/SCAL2をサポート可能、SCAL2では2要素認証が必要
- **Explicit authorization**（明示的認可）
  - ✓ 署名アプリケーションを**経由して**SAMが利用者と認可プロセスを管理
  - ✓ PINやOTPを使ったCSC仕様にてSCAL1/SCAL2をサポート可能
  - ✓ CSC仕様を使う場合にはExplicit Authorizationを使うと楽と思われる
- **OAuth 2.0 authorization**（RFC 6749）
  - □ 詳しくは後述（ただクレデンシャル認可に使うのは面倒では？）

※ **SCAL：Sole Control Assurance Level ⇒ CEN 419 241-1**
低レベルのSCAL1と、高レベルのSCAL2の2レベルがある

疑問：SCAL2の2要素認証の1つにサービス認証を含んで良いか？おそらくダメか？
ダメだとするとSAMで2要素を管理する必要がある。**※今後確認が必要。**

関連：CSC_0.1.7.9_PR ページ 13

クレデンシャル認可レベル（SCAL：単独制御保証レベル）

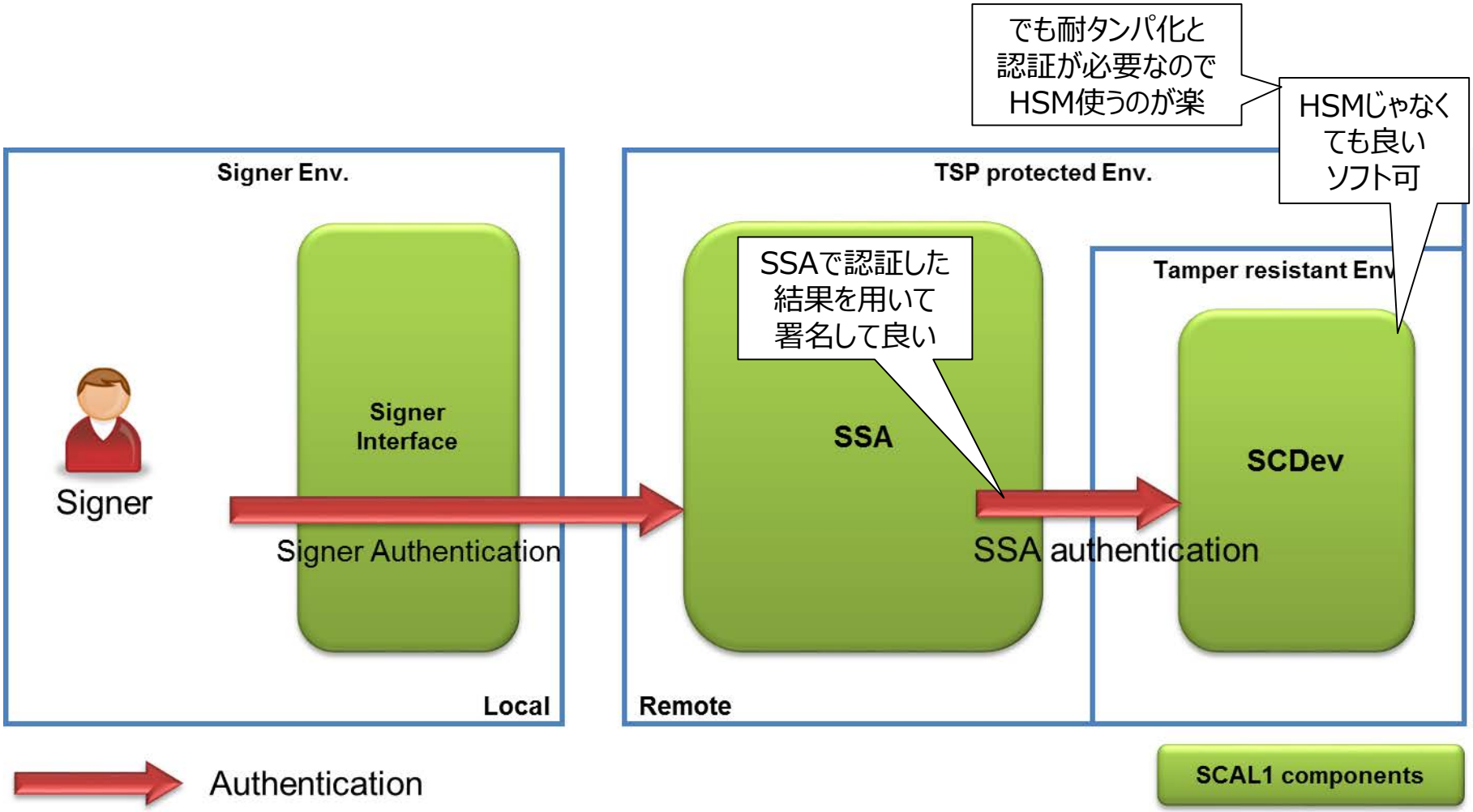
※ **CEN 419 241-1の定義**（CEN仕様書は有償なので注意）

- **Sole control assurance level 1 (SCAL1):**
  Non Qualified Level（１国内の電子取引に有効なレベル）
  - ✓ 基本的な（署名アプリ側の間接的な）認証があれば、SADとSAMは使わなくても良い（SADとはSAM発行のトークンであり、使っても良い）
    ※ SAD: Signature Activation **Data** / SAM: Signature Activation **Module**
  - ✓ ハードウェアを使わずソフトウェアで実現しても良い

- **Sole control assurance level 2 (SCAL2):**
  Qualified Level（国を跨った電子取引でも有効なレベル）
  - ✓ SAM発行のSADと、SAM管理の2要素認証メカニズムが必要
  - ✓ 耐タンパ領域にSAMと鍵管理（HSM）を持つハードウェアで実現
    ※ CEN/FIPS等の標準化されたセキュリティ基準への準拠が必要
  - ✓ eIDAS規制のスタンドアロンQSCDと同じレベルを要求。
    ※ QSCD（**Qualified** electronic Signature Creation Device）

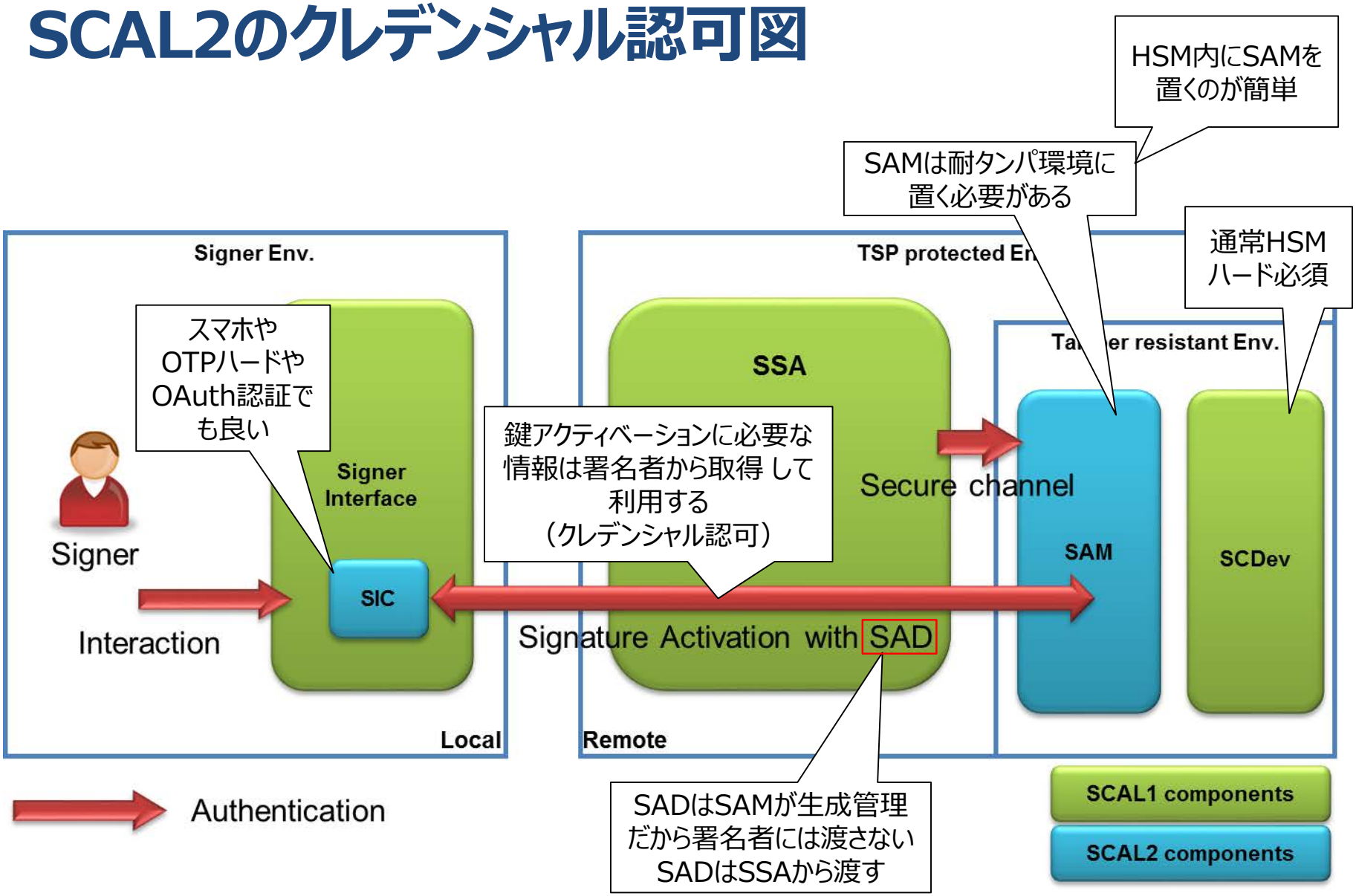
（参考情報）

SCAL1のクレデンシャル認可図



SCAL1だとSAM/SADとSICは使わなくても良い。

SCAL2のクレデンシャル認可図



OAuth 2.0 認証（RFC 6749）

サービスとクレデンシャルの認証/認可に利用が可能。

- scopeパラメーターで区別：service | credential

現在以下の3種類の認証フローをサポート：

- **Authorization Code flow**（サービスとクレデンシャル）
  WebアプリのようにClient Secretを安全に保持可能な場合の**最も推奨されるフロー**
- **Implicit Grant flow**（サービスとクレデンシャル）
  Client Secretを安全に保持できないスマホアプリやJavaScriptの場合に推奨されるフロー
- **Client Credentials flow**（サービスのみ）
  社内のAPI等においてOauthを使う時のフロー（外部サービスでの利用は難しい）

詳しくはOAuth 2.0の仕様書を確認すると良い。
https://openid-foundation-japan.github.io/rfc6749.ja.html

※ バイオメトリック認証や電話を利用した認証もサポート

関連：CSC_0.1.7.9_PR ページ 13

CSCのAPI一覧

API	概要
info	リモートサービスの実装APIメソッドのリスト情報を返す
auth/login	Basic認証/Digest認証による認証（access_tokenの取得）
auth/revoke	access_token または refresh_token の取り消し
oauth2/authorize **※ OAuth 2.0 の外部API**	OAuth 2.0のAuthorization Code FlowかImplicit Grant Flowによる認証 scope=credential なら取得したaccess_tokenをSADとして利用
oauth2/token	OAuth 2.0 の access_token を返す（Authorization Code Flowやrefresh）
credentials/list	ユーザーに関連付けられたクレデンシャル情報のリストを返す
credentials/info	署名クレデンシャル情報を返す（鍵、証明書やPIN/OTP等の認可方法等）
credentials/authorize	署名する為のクレデンシャル認可（PIN/OTPによる認証）SADの取得
credentials /extendTransaction	複数署名を連続して行う場合のトランザクション拡張 新しいSAD（Signature Activation Data）の取得
credentials/sendOTP	クレデンシャルに関連付いたオンラインOTP機能の開始（メール、SMS等）
signatures/signHash	ハッシュへの署名実行（**※SADが必要**）
signatures/timestamp	指定ハッシュ値からタイムスタンプトークンの取得（長期署名用でSAD不要）

関連：CSC_0.1.7.9_PR ページ　14-15,24

info sample

Request

```
POST https://service.domain.org/csc/v0/info
```

Response

```
HTTP/1.1 200 OK
{
   "specs": "1.1",
   "name": "ACME Trust Services",
   "logo": "https://service.domain.org/images/logo.png",
   "region": "IT",
   "lang": "en-US",
   "authType": ["basic", "oauth2code", "oauth2implicit"],
   "oauth2": "https://www.domain.org/oauth2/authorize",
   "methods":
   [
      "auth/login",
      "auth/revoke",
      "oauth2/token",
      "credentials/list",
      "credentials/info",
      "credentials/authorize",
      "credentials/sendOTP",
      "signatures/signHash"
   ]
}
```

logoも必須、JPEG/PNGで256x256以下

external, TLS, basic, digest, oauth2code, oauth2implicit, oauth2client

← 使える認証方式

← OAuth2のIdPのURI

使えるCSCのAPI

認証不要で情報を取得可能。仕様上はPOSTだけどGETでも取得できた方が良いよね？

関連：CSC_0.1.7.9_PR ページ 25-26

auth/login sample

Request	POST https://service.domain.org/csc/v0/auth/login
	Authorization: Basic Y2xpZW5OX2lkOmNsaWVudF9zZWNyZXQ=
	Content-Type: application/json
	{
	"rememberMe": true
	}

Response	HTTP/1.1 200 OK
	{
	"access_token": "4/CKN69L8gdSYp5_pwH3XlFQZ3ndFhkXf9P2_TiHRG-bA",
	"refresh_token": "_TiHRG-bA H3XlFQZ3ndFhkXf9P24/CKN69L8gdSYp5_pw",
	"expires_in": 3600
	}

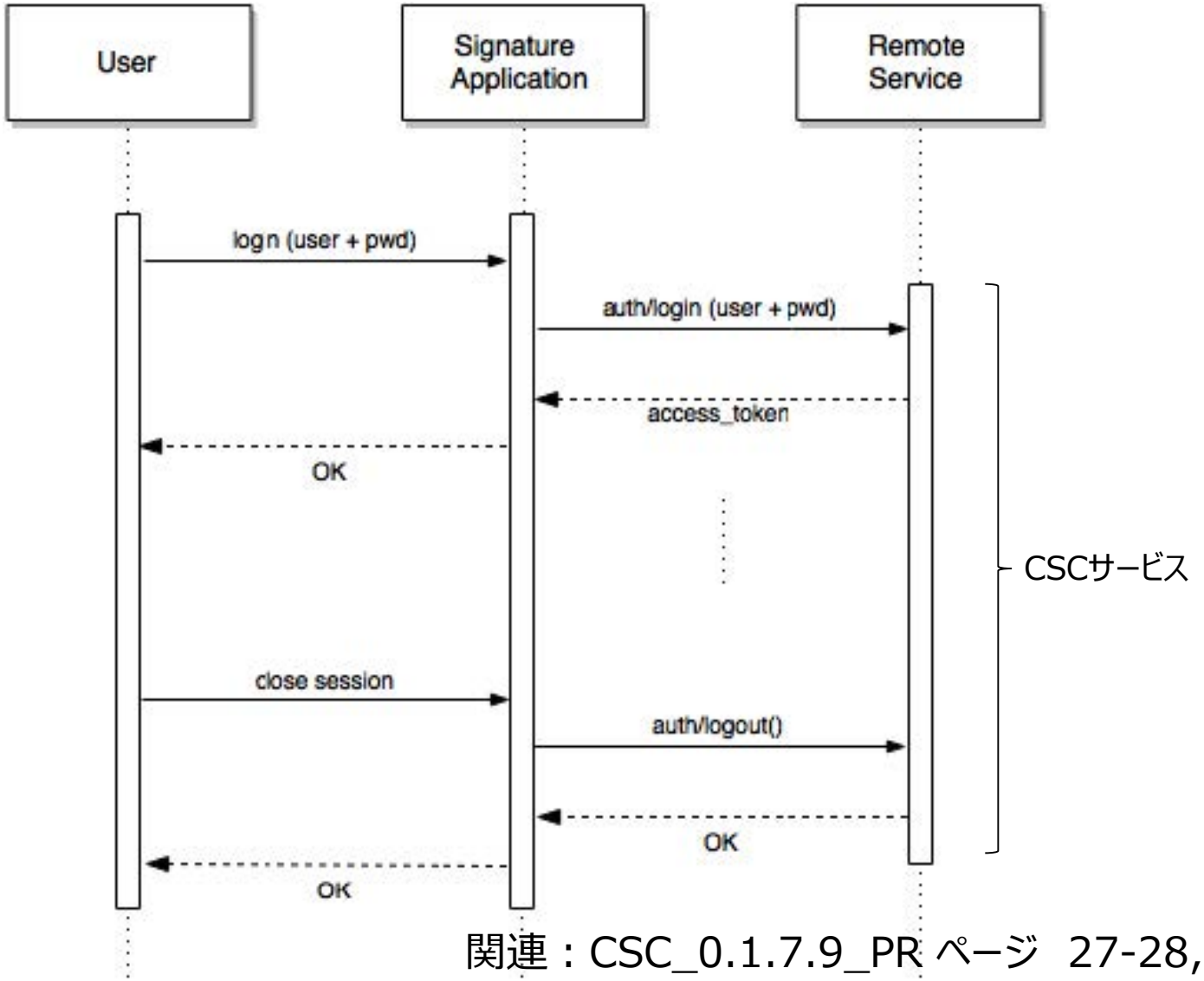
Basic認証・Digest認証の実装用API。
access_token（Bearer認証）に対応する必要がある。

関連：CSC_0.1.7.9_PR ページ 27-28

サービス認証：Basic認証/Digest認証

RSSP service authorization using a username and password

- シンプル！
- IdP連携は不可。
- アクセストークンを返す必要あり。



関連：CSC_0.1.7.9_PR ページ 27-28,49

oauth2/authorize (Authorization Code) sample

Request

```
GET https://www.domain.org/oauth2/authorize?response_type=code&
 client_id=[CLIENT_ID]&redirect_uri=[REDIRECT_URI]&scope=service&state=12345678&
 lang=en-US&appName=Cloud%20Signature%20Service
```

Response

```
HTTP/1.1 302 Found
Location: [REDIRECT_URI]?code=FhkXf9P269L8g&state=12345678
```

Authorization Codeを使う場合、認証時にサービス側へ接続して許可を得ると共にコードを共有しておく。

oauth2/token (Authorization Code) sample

Request

```
POST https://service.domain.org/csc/v0/oauth2/token
Content-Type: application/json
{
   "grant_type": "authorization_code",
   "code": "FhkXf9P269L8g",
   "client_id": "test",
   "client_secret": "password"
}
```

OAuth2.0の仕様だとtokenエンドポイントは認可サーバ側で用意。

事前登録

authorization_code
client_credentials
refresh_token
scope=serviceのみ

Response

```
HTTP/1.1 200 OK
{
   "access_token": "4/CKN69L8gdSYp5_pwH3XIFQZ3ndFhkXf9P2_TiHRG-bA",
   "refresh_token": "_TiHRG-bA H3XIFQZ3ndFhkXf9P24/CKN69L8gdSYp5_pw",
   "token_type": "Bearer",
   "expires_in": 3600
}
```

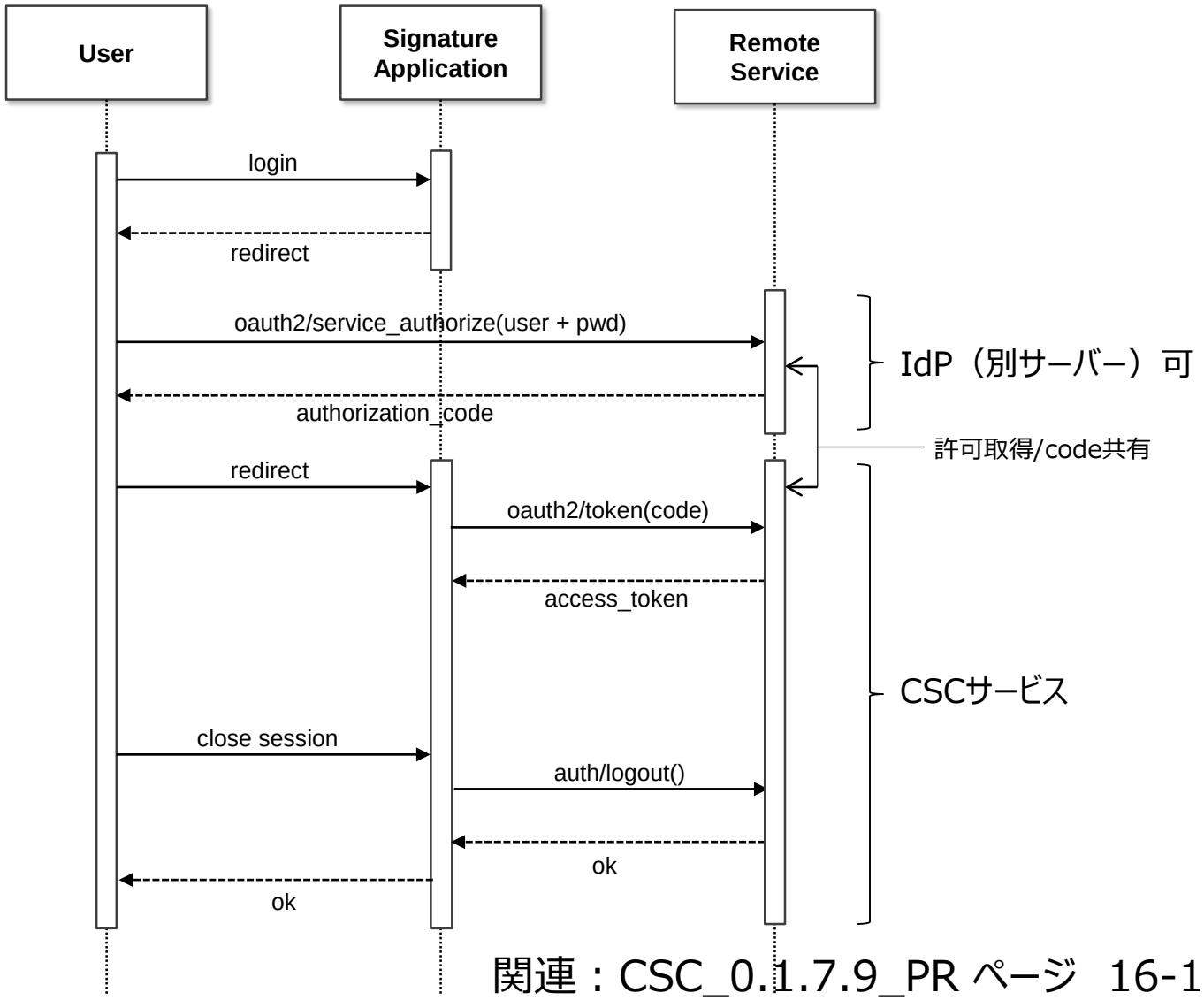
ベアラトークンとは購入済みチケットのようなもの

関連：CSC_0.1.7.9_PR ページ 16-18

認証：OAuth2 Code Flow 認証

RSSP service authorization using OAuth2 with Authorization Code flow

- Authorization Code Flow はWebサービス向けの手順。
- access_token が User に渡らないので望ましい。
- OAuth 2.0 を使うなら Authorization Code Flow を使うべきだろう。



関連：CSC_0.1.7.9_PR ページ 16-18

oauth2/authorize (Implicit Flow / service) sample

Request	GET https://www.domain.org/oauth2/authorize?response_type=token& client_id=[CLIENT_ID]&redirect_uri=[REDIRECT_URI]&scope=service& state=12345678&lang=en-US&appName=Cloud%20Signature%20Service
Response	HTTP/1.1 302 Found Location: [REDIRECT_URI]?token_type=Bearer& access_token=4/CKN69L8gdSYp5_pwH3XlFQZ3ndFhkXf9P2_TiHRG-bA& expires_in=3600&state=12345678

oauth2/authorize (Implicit Flow / credential) sample

Request	GET https://www.domain.org/oauth2/authorize?response_type=token& client_id=[CLIENT_ID]&redirect_uri=[REDIRECT_URI]&scope=credential& state=12345678&credentialID=GX0112348&numSignatures=1 Authorization: Bearer 4/CKN69L8gdSYp5_pwH3XlFQZ3ndFhkXf9P2_TiHRG-bA
Response	HTTP/1.1 302 Found Location: [REDIRECT_URI]?access_token=FhkXf9P269L8g&token_type=Bearer& expires_in=3600&state=12345678

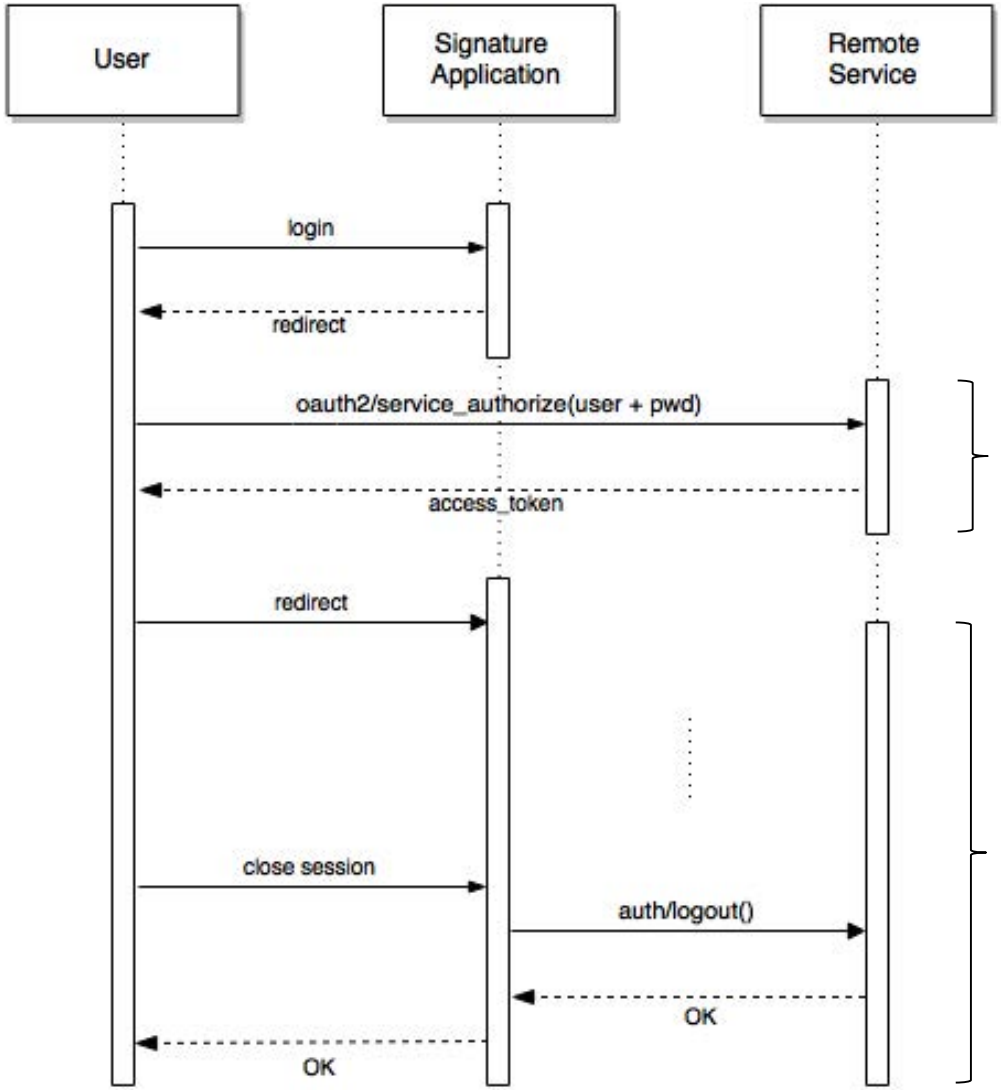
credential では取得した access_token を SAD として利用。

implicit（暗黙的）認証では直接署名用のアクセストークンを取得
クライアントの認証は行われない為にURL等の事前登録が必須である
リフレッシュトークンはサポートされない

関連：CSC_0.1.7.9_PR ページ 19-21

認証：OAuth2 Implicit Flow 認証

RSSP service authorization using OAuth2 with Implicit Grant flow



- スマホアプリやJavaScript向きの手順（秘密コードが守れないケース）。
- access_token が User を経由してしまう。
- CSC利用を想定するなら Authorization Code Flow の方が望ましいと考える。

IdP（別サーバー）可

CSCサービス

関連：CSC_0.1.7.9_PR ページ 16-18,50

credentials/list sample

Request	POST https://service.domain.org/csc/v0/credentials/list **Authorization: Bearer** 4/CKN69L8gdSYp5_pwH3XlFQZ3ndFhkXf9P2_TiHRG-bA

Response	HTTP/1.1 200 OK
	{
	"credentialIDs":
	[
	"GX0112348",
	"HX0224685"
	]
	}

認証されたユーザの利用可能な鍵（クレデンシャル）のリスト。

関連：CSC_0.1.7.9_PR ページ 34-35

credentials/info sample

Request

```
POST https://service.domain.org/csc/v0/credentials/info
Authorization: Bearer 4/CKN69L8gdSYp5_pwH3XIFQZ3ndFhkXf9P2_TiHRG-bA
Content-Type: application/json
{
   "credentialID": "GX0112348",  "certificates": "chain",
   "certInfo": true, "authInfo": true
}
```

certificates:
"none", "single", "chain"

certInfo（証明書情報）と、
authInfo（認証情報）は、
個別にオン・オフが可能。

Response

```
HTTP/1.1 200 OK
{
   "key":
   {
      "status": "enabled",
      "algo": [ "1.2.840.113549.1.1.1", "0.4.0.127.0.7.1.1.4.1.3" ],
      "len": 2048
   },
   "cert":
   {
      "status": "valid",
      "certificates":
      [
         "Base64-encoded X.509 end entity certificate",
         "Base64-encoded X.509 intermediate CA certificate",
         "Base64-encoded X.509 issuer CA certificate"
      ],
      "issuerDN": "The X.500 issuer DN printable string",
```

status:(サンプルでは"active"になっているが間違いか？)
"enabled", "disabled"

鍵長　RSA encryption　ecdsa-plain-SHA256

status:
"valid", "expired", "revoked", "suspended"

証明書チェーン

関連：CSC_0.1.7.9_PR ページ 36-40

credentials/info sample (continue)

Response (continue)

```
        "serialNumber": "5AAC41CD8FA22B953640",
        "subjectDN": "The X.500 subject DN printable string",
        "validFrom": "20160101100000Z",
        "validTo": "20190101095959Z"
    },
    "authMode": "explicit",
    "PIN":
    {
        "presence": true
        "label": "PIN",
        "description": "Please type the signature PIN"
    },
    "OTP":
    {
        "presence": true,
        "type": "offline",
        "ID": "MBO1-K741200",
        "provider": "totp",
        "format": "N",
        "label": "Mobile OTP",
        "description": "Please type the 6 digit code you received on your phone"
    },
    "multisign": true, ←―― 複数署名
    "lang": "en-US"
}
```

GeneralizedTime Format (RFC 2459)

クレデンシャル認可のモード：
- implicit: リモート内で独立して管理（署名アプリを経由しない）
- explicit: 署名アプリ経由可で2つ以上のセキュリティ要素を利用
- oauth2code: OAuth 2.0 authoritzation code
- oauth2token: OAuth 2.0 implicit grant

関連：CSC_0.1.7.9_PR ページ 36-40

credentials/authorize sample

Request

```
POST https://service.domain.org/csc/v0/credentials/authorize
Authorization: Bearer 4/CKN69L8gdSYp5_pwH3XIFQZ3ndFhkXf9P2_TiHRG-bA
Content-Type: application/json
{
   "credentialID": "GX0112348",
   "numSignatures": 2,
   "hash":
   [
      "sTOgwOm+474gFjOqOx1iSNspKqbcse4IeiqlDg/HWuI=",
      "c1RPZ3dPbSsONzRnRmowcTB4MWITTnNwS3FiY3NINEllaXFsRGcvSFd1STO="
   ],
   "PIN": "12345678",
   "OTP": "738496",
   "clientData": "12345678"
}
```

`numSignatures`: 署名数もSCLA2の時に利用 ※ 署名数を管理

`hash`: 署名対象ハッシュ値

`PIN`, `OTP`: クライアントで取得したPINとOTPの値を指定

Response

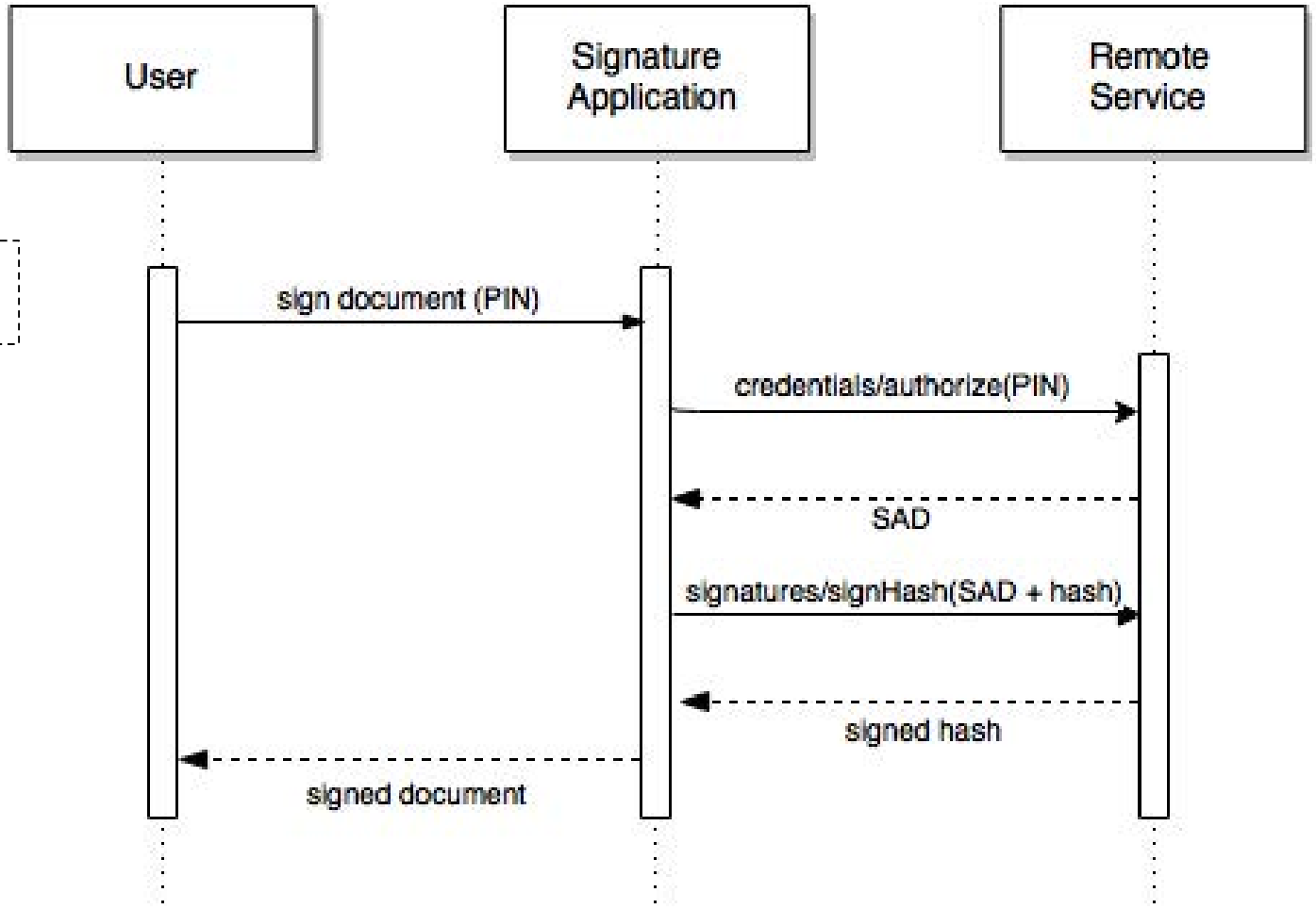
```
HTTP/1.1 200 OK
{
   "SAD": "_TiHRG-bAH3XIFQZ3ndFhkXf9P24/CKN69L8gdSYp5_pw"
}
```

PINやOTPによるクレデンシャル認可時に利用されるAPI（OAuth2では使ってはいけない）
hashパラメーターは署名する対象を指定することで不正アクセスを防ぐと思うが説明がない
credentials/infoのSCLAが"1"なら呼び出し不要との記述あり（ページ37）

関連：CSC_0.1.7.9_PR ページ 41-42

クレデンシャル認可：PIN/OTP/PIN+OTP

Create a remote signature with a credential protected by a PIN



- OTPのみとPIN+OTPも基本的に同じ手順

関連：CSC_0.1.7.9_PR ページ 41-42,52-55

credentials/sendOTP sample

Request	POST https://service.domain.org/csc/v0/credentials/sendOTP
	Authorization: Bearer 4/CKN69L8gdSYp5_pwH3XIFQZ3ndFhkXf9P2_TiHRG-bA
	Content-Type: application/json
	{
	"credentialID": "GX0112348",
	"clientData": "12345678"
	}

Response	HTTP/1.1 200 OK

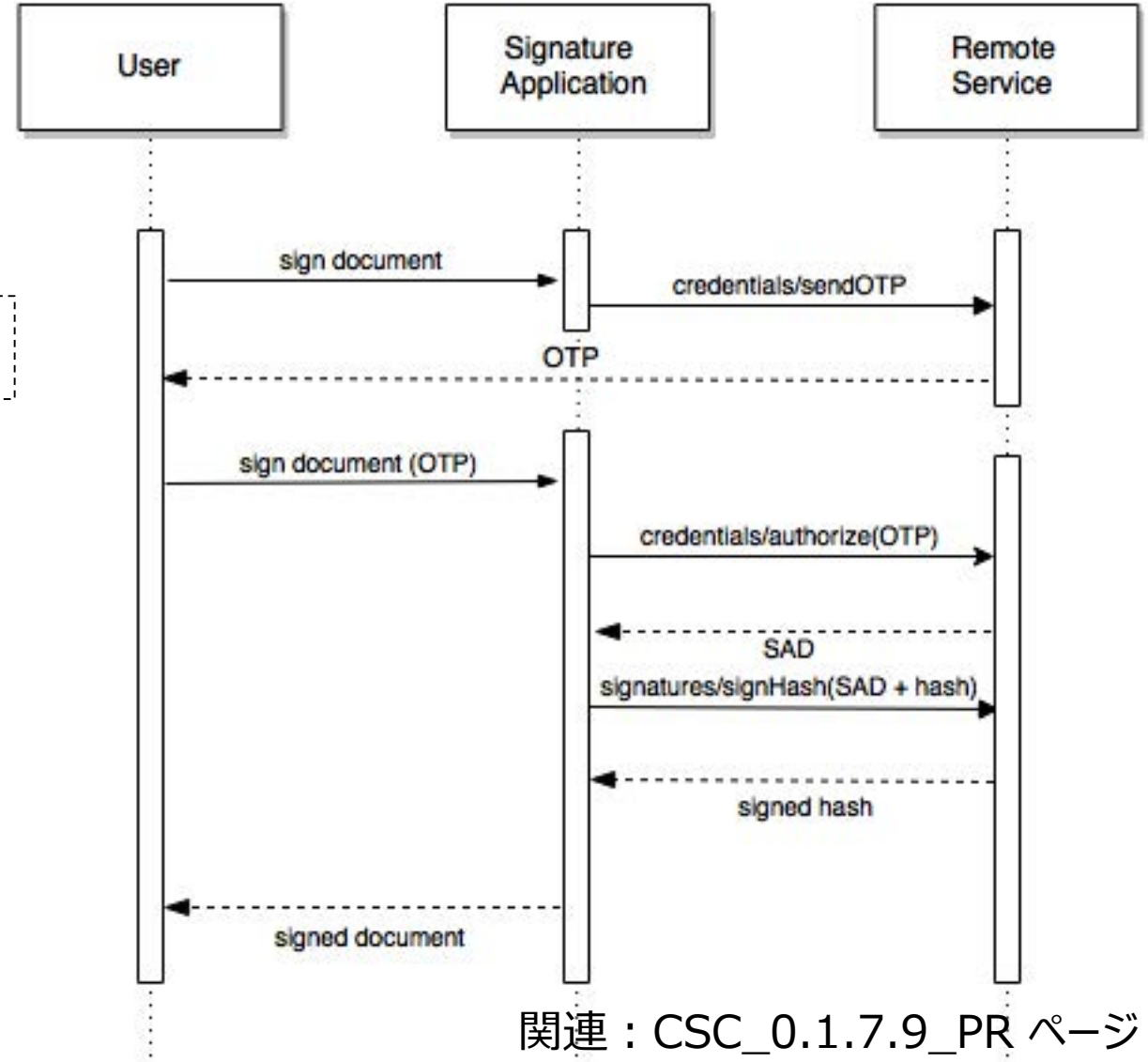
オンラインでOTPを生成してSMSやメール等の別ルートでユーザに送信する為のAPI。
※ OTP：One Time Password

関連：CSC_0.1.7.9_PR ページ 44

クレデンシャル認可：オンラインOTP

Create a remote signature with a credential protected by an “online” OTP

- 生成したOTPはSMSやメール等の別経路でUserに渡す。

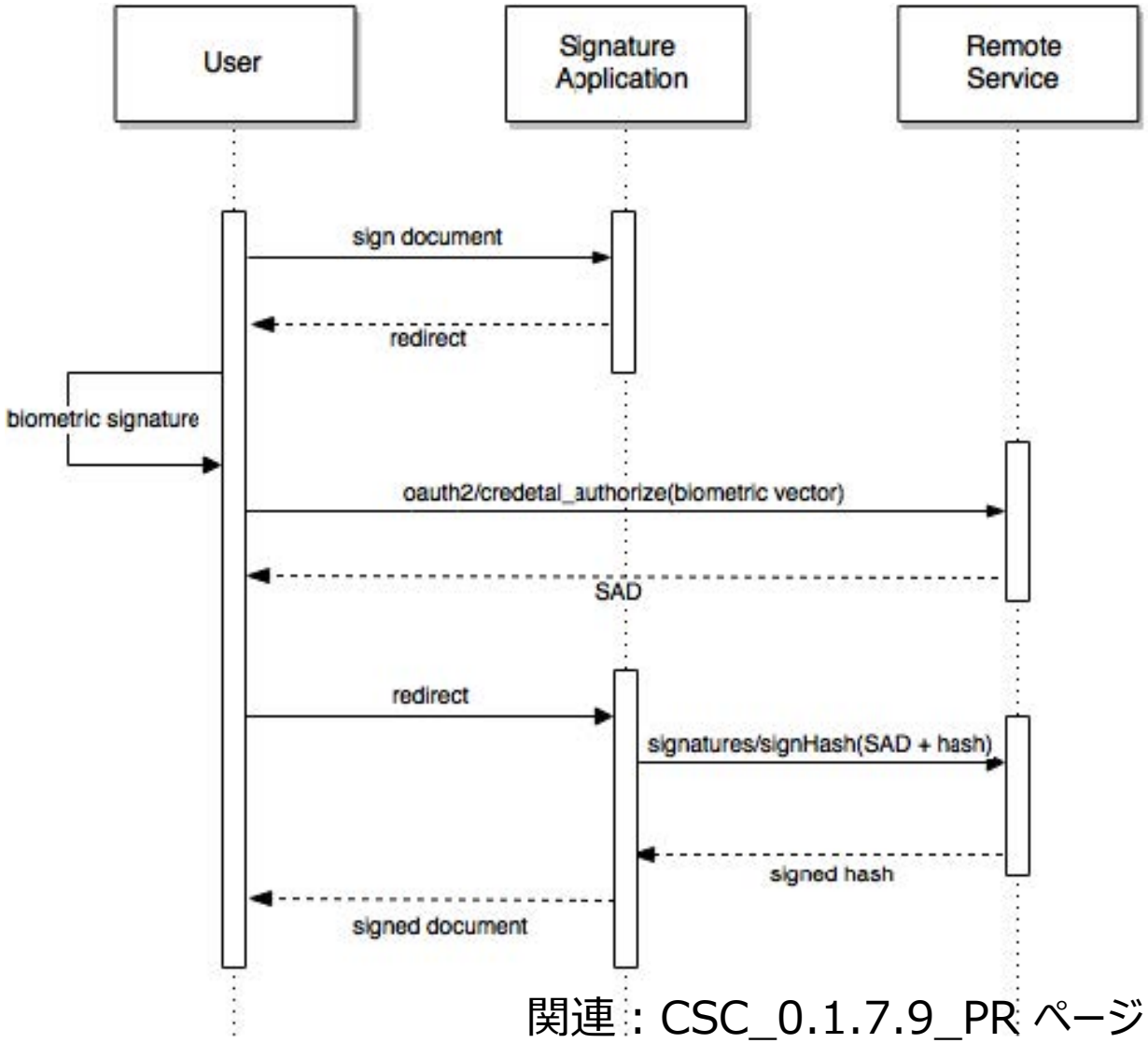


関連：CSC_0.1.7.9_PR ページ 44,54

クレデンシャル認可：バイオ認証

Create a remote signature with a credential protected by a biometric factor

- 参考用。実際に使われる可能性はあるかな？



関連：CSC_0.1.7.9_PR ページ 56

signatures/signHash sample

Request

```
POST https://service.domain.org/csc/v0/signatures/signHash
Authorization: Bearer 4/CKN69L8gdSYp5_pwH3XIFQZ3ndFhkXf9P2_TiHRG-bA
Content-Type: application/json
{
   "credentialID": "GX0112348",
   "SAD": "_TiHRG-bAH3XIFQZ3ndFhkXf9P24/CKN69L8gdSYp5_pw",
   "hash":
   [
      "sTOgwOm+474gFjOqOx1iSNspKqbcse4IeiqlDg/HWuI=",
      "c1RPZ3dPbSsONzRnRmowcTB4MWITTnNwS3FiY3NINEllaXFsRGcvSFd1STO="
   ],
   "hashAlgo": "2.16.840.1.101.3.4.2.1",
   "signAlgo": "1.2.840.113549.1.1.1",
   "clientData": "12345678"
}
```

署名対象

Response

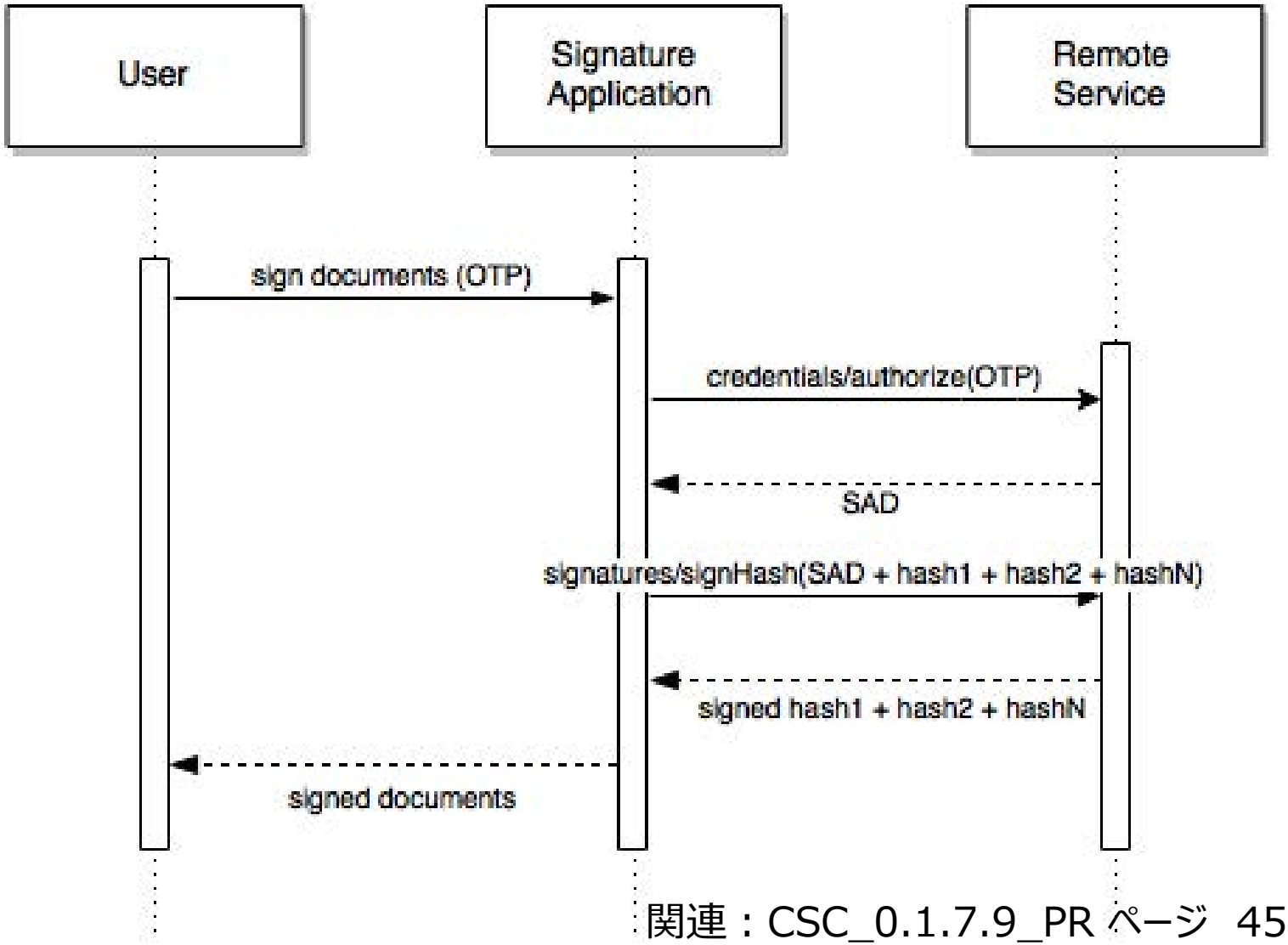
```
HTTP/1.1 200 OK
{
   "signatures":
   [
      "KedJuTob5gtvYx9qM3k3gm7kbLBwVbEQRl26S2tmXjqNND7MRGtoew==",
      "Idhef7xzgtvYx9qM3k3gm7kbLBwVbE98239S2tm8hUh85KKsfdowel=="
   ]
}
```

署名値

関連：CSC_0.1.7.9_PR ページ 45-46

複数署名：一括（1回のAPIで複数署名計算）

Create multiple remote signatures from a list of hash values



関連：CSC_0.1.7.9_PR ページ 45-46,57

credentials/extendTransaction sample

Request	POST https://service.domain.org/csc/v0/credentials/extendTransaction
	Authorization: Bearer 4/CKN69L8gdSYp5_pwH3XIFQZ3ndFhkXf9P2_TiHRG-bA
	Content-Type: application/json
	{
	"credentialID": "GX0112348",
	"SAD": "_TiHRG-bAH3XIFQZ3ndFhkXf9P24/CKN69L8gdSYp5_pw",
	"clientData": "12345678"
	}

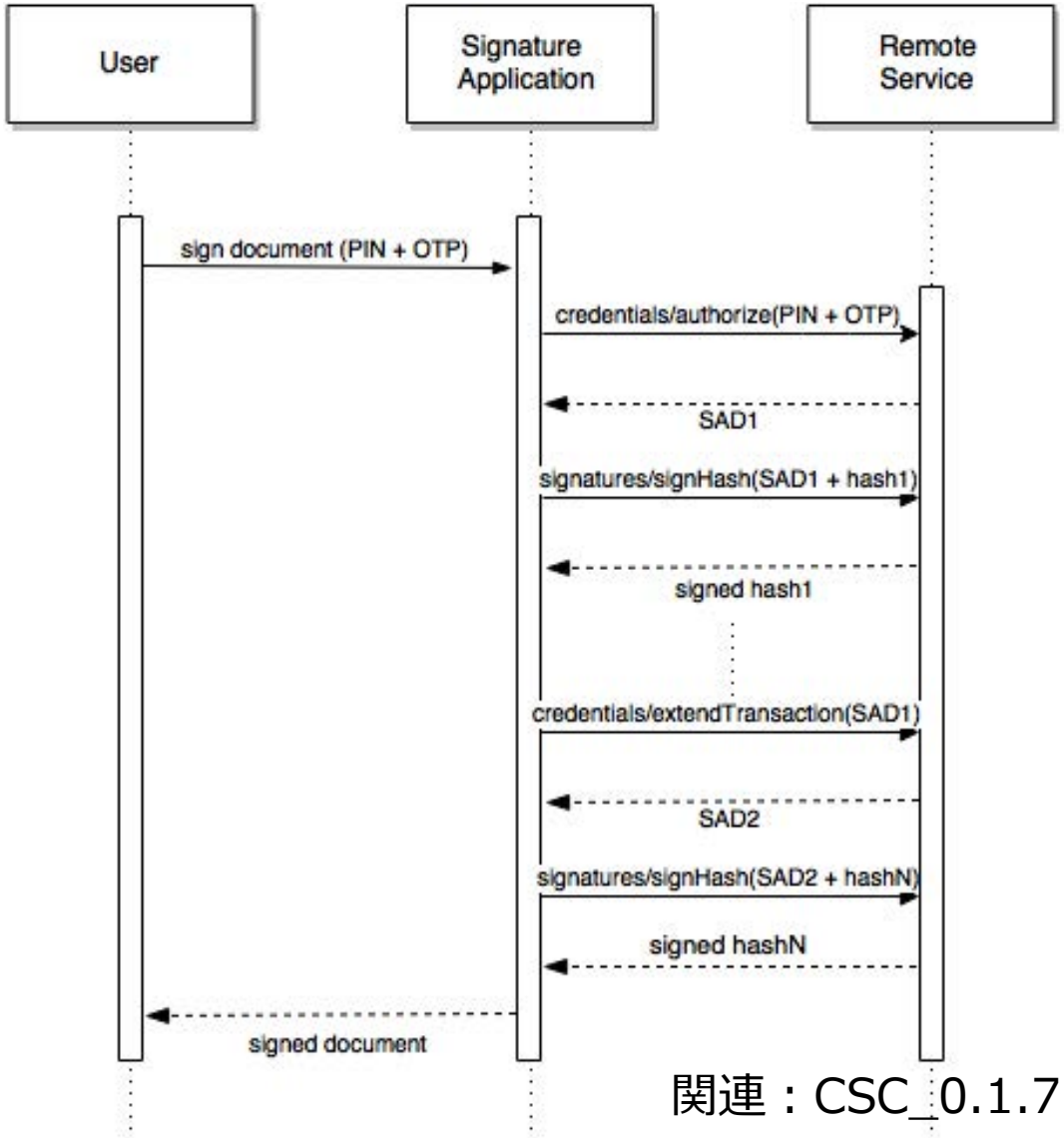
Response	HTTP/1.1 200 OK
	{
	"SAD": "1/UsHDJ98349h9fgh9348hKKHDkHWVkI/8hsAW5usc8_5=",
	}

使い終わったSADから新しいSADを取得するAPI。

関連：CSC_0.1.7.9_PR ページ 43

複数署名：連続（複数の署名APIの連続呼び出し）

Create a remote multi-signatures transaction



関連：CSC_0.1.7.9_PR ページ 43,58

signatures/timestamp sample

Request

```
POST https://service.domain.org/csc/v0/signatures/timestamp
Authorization: Bearer 4/CKN69L8gdSYp5_pwH3XIFQZ3ndFhkXf9P2_TiHRG-bA
Content-Type: application/json
{
   "hash": "sTOgwOm+474gFjOqOx1iSNspKqbcse4IeiqlDg/HWuI=",
   "hashAlgo": "2.16.840.1.101.3.4.2.1",
   "clientData": "12345678"
}
```

（Authorization: Bearer の行：サービス認証で取得したtoken）
（"hash" の行：署名値のハッシュ）
（"hashAlgo" の行：SHA-256）

Response

```
HTTP/1.1 200 OK
{
   "timestamp": "MGwCAQEGCSsGAQQB7U8CATAxMA0･･･EzMjM5WjADAgEBAgkAnWn2SSIWIXk="
}
```

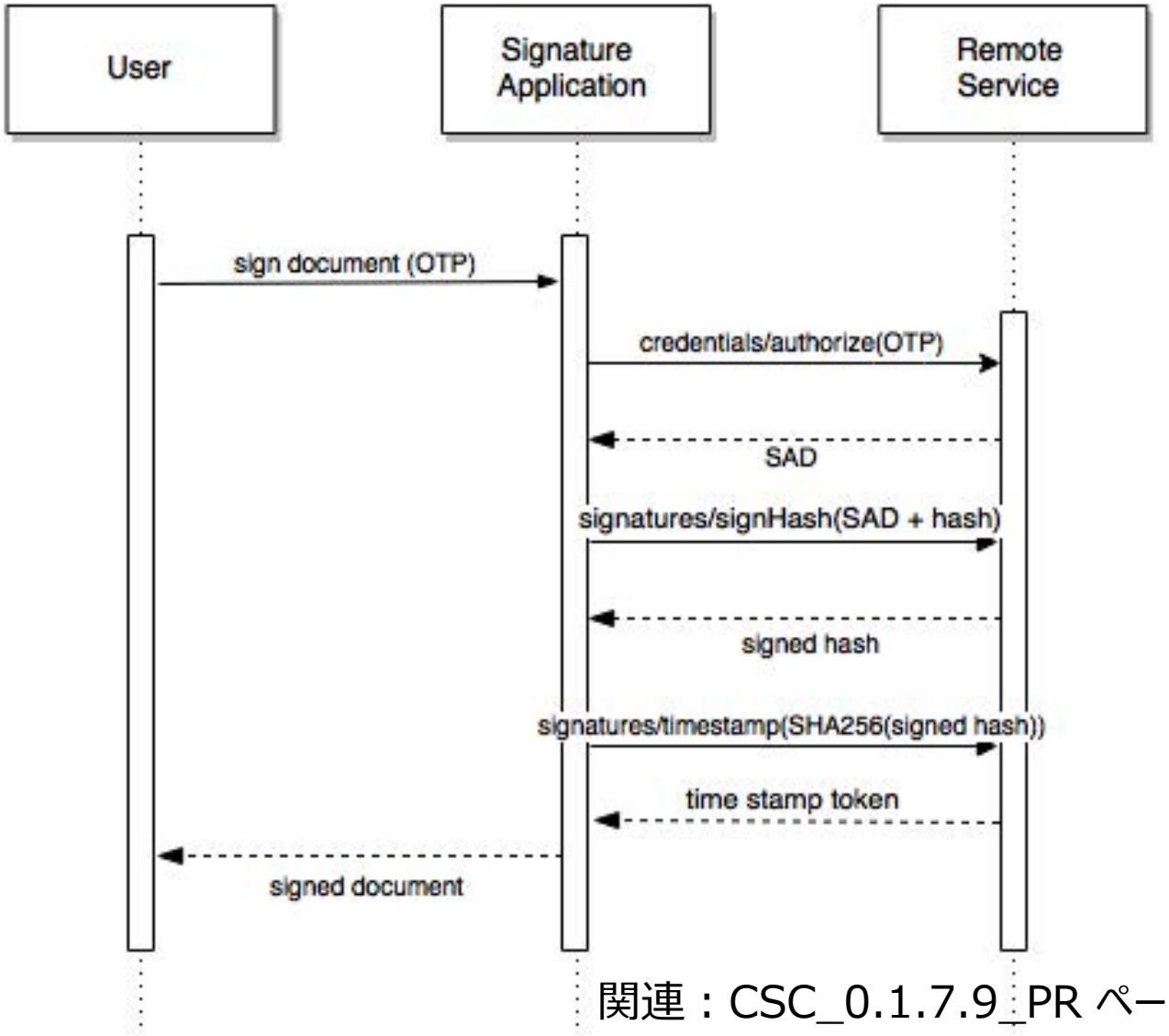
（"timestamp" の行：タイムスタンプトークン）

SAD等のクレデンシャル認可は不要だがサービス認証は必要。
長期署名の署名タイムスタンプ・アーカイブタイムスタンプ用と考えている様子。
タイムスタンプ単独での利用は想定していないようだ。
※ もちろんタイムスタンプのみ利用しても良いはず

関連：CSC_0.1.7.9_PR ページ 47-48

長期署名：署名＋タイムスタンプ

Create a remote signature with long-term validity profile



関連：CSC_0.1.7.9_PR ページ 47-48,59