Smart-in・企業サーバ間
インタフェース仕様書
PreCall 機能

2017/9/20

第 1.4 版

株式会社あいびし

改訂履歴

Ver.	年月日	内容	備考
1.0	2015/11/19	初版リリース	
1.1	2017/6/9	P1　タイムアウトシーケンスは存在しない為、削除。 P2　送信例を追記。 P3　送受信ライブラリ→認証コード生成ライブラリ 　　キーファイルに関する記述の削除。	
1.2	2017/6/21	P2　不要な記述を削除	
1.3	2017/7/18	P5　サンプルコードを追記	
1.4	2017/9/20	タイトル修正	

目次

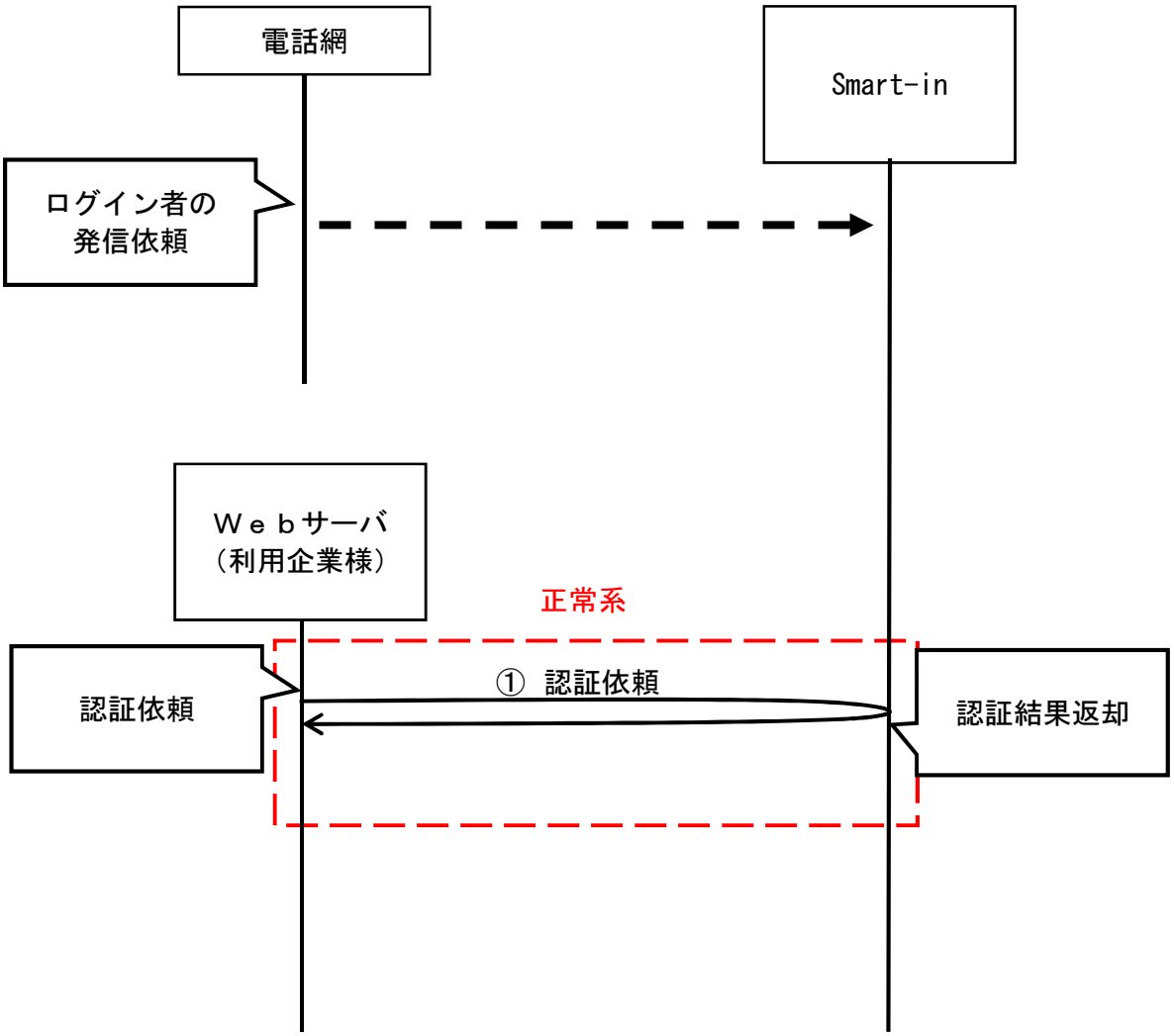
1. 本書の目的

本書では、Smart-in と企業サーバ間のインタフェース仕様（PreCall 機能）を記述いたします。

2. 処理フロー

電話依頼～認証完了までは以下のフローとします。

2.1. 先行発信認証方式



※ 通信プロトコルは HTTPS を使用します。

インタフェース定義

2.2. 共通

・文字コード：UTF8

2.3. 先行発信認証依頼

・リクエスト方向

利用企業様　→　Smart-in

以下の内容で HTTPS 送信してください。

URL	https:// {接続サーバドメイン名} /request.cgi	
送信タイプ	POST	
パラメータ	pubkey	企業キー Smart-in システムから提示する 32 桁の文字列
	authcode	電話番号をハッシュ化した文字列

・リクエストパラメータ

論理名	物理名	JSON 内 論理名	必須	説明
企業キー	pubkey	キー	○	32 桁の文字列
依頼データ	authcode	認証コード	○	電話番号を後述するライブラリでハッシュ化した文字列 コマンド名　authcode_crypt

・レスポンス

論理名	物理名	暗号化	説明
結果	result	-	正常：0 ／ 異常：9
リターン詳細	detail	-	異常時の理由 ※ 正常時 Null 11：依頼データで指定された電話番号からの着信無 or 企業キーが不正。 12：Smart-in システムビジー

レスポンス例：正常

```
{"result":"0","detail":nul}
```

レスポンスには改行が含まれません。

レスポンス例：異常（依頼データの番号から着信を受けていない場合）

```
{"result":"9","detail":"11"}
```

3. 認証コード生成ライブラリ

3.1. 機能説明

Smart-in への先行発信認証依頼の依頼データを生成します。

3.2. 提供形式

以下のファイルを提供いたします。

モジュール名	説明
authcode_crypt	先行発信認証方式で電話番号から認証コードを生成するモジュールです。 ※Windows 版は、authcode_crypt.exe

3.3. パラメータ形式

3.3.1. authcode_crypt 使用時

ファイル名	authcode_crypt [target_string]	
パラメータ	第 1 引数	target_string：エンコードする電話番号
	リターン値	エンコードされた文字列

実行例：

コマンド

./authcode_crypt 09012345678

実行結果

d13230b9a9f9fd81f897b2da29a8b1a197bc20062c1022827f34f1b8a7a60196

4. サンプルコード

本プログラムコードは 3 ファイルで構成しており、以下にその抜粋を掲載いたします。

4.1. index.html（一部抜粋）

処理概要

・ログインフォームを表示します。
・認証ボタン押下時、認証コード（電話番号のハッシュ）を取得します。
・取得した認証コードを基に、認証結果の取得を行います。認証が OK になった場合、別ページにリダイレクトします。

※jQuery を使用しています。

```
・・・略・・・

<form>
<input type="text" id="telno" />
<btn id="btn_auth" role="button">認証</btn>
</form>
<span id="progress_txt"> </span>

・・・略・・・

<script type="text/javascript">
    // 発信先電話番号
    var precall_telno = "050-1111-2222";
    // smart-in 本体の URL（相対パス）
    var auth_url = "./getauthresult.php";
    // ログイン成功時 URL
    var menu_url = "./menu.html";

    $(function(){
        ////////////////////////
        // 認証ボタン押下時
        $('#btn_auth').click(function(){

            var telno = $('#telno').val().replace(/-/g, '');

            // 電話番号のハッシュ情報取得
            $.ajax({
                url: "getauthcode.php",
                data: {
                    telno: telno,
                },
                type: "GET",
                cache: false,
                success: function (data) {
                    authenticate(data);
                }
            });
        });
    });

・・・
```

次ページへ続く

・・・前ページの続き

```
    /**
     * 認証をおこなう
     */
    function authenticate(authcode){
        var auth_cnt = 15;  // 認証のリトライを行う回数

        var timer = setInterval(function(){
            $.ajax({
                url: auth_url,
                data: {
                    authcode : authcode
                },
                type: "POST",
                dataType: 'json',
                xhrFields: { withCredentials: true },
                cache: false,
                success: function(json){
                    if(json.result == "0"){
                        // 認証 OK 時メニュー画面へリダイレクト
                        location.href= menu_url;
                        clearInterval(timer);
                        return;
                    }
                    else{
                        //if(json.result == "9"){
                        auth_cnt-=1;
                        if(auth_cnt == 0){
                            // 認証タイムアウト
                            $("#progress_txt").html("<p class='big'>認証がタイムアウトしました。<br /><a
href='./'>再ログイン</a></p>");
                            clearInterval(timer);
                            return;
                        }
                        $("#progress_txt").append("・");
                        //}
                    }
                },
                error: function () {
                    alert('接続に失敗しました。');
                    clearInterval(timer);
                    return;
                }
            });
        },2000);   // 2 秒毎にサーバへ問合せをする（リトライ回数と併せて、この場合 30 秒待つこととなる）
    }
</script>
```

4.2. getauthcode.php

処理概要

・電話番号を独自のルールでハッシュ化し、ブラウザに返します。

```
<?php
$telno = $_GET['telno'];

// 電話番号のハッシュデータ取得
// サーバ内の任意の場所に設置した authcode_crypt を実行し、実行結果を取得する
$authcode = exec("/usr/lib/authcode_crypt ".$telno);

echo $authcode;
```

4.3. getauthresult.php

処理概要

・企業キーと認証コード（電話番号のハッシュ）を基に、Smart-in サーバに認証結果を問い合わせます。
・結果をブラウザに返します。

```
<?php

// sin_crypt 暗号化キー（半角英数 32 桁）
define('PUB_KEY', '1234567890abcdef1234567890abcdef');
// smart-in の URL
define('SMARTIN_URL', 'https://********.***/request.cgi');

$authcode = $_POST['authcode'];

$postdata = http_build_query(
    array(
        'pubkey' => PUB_KEY,
        'authcode' => $authcode
    )
);

// http 設定
$opts = array('http' =>
    array(
        'method'  => 'POST',
        'header'  => 'Content-type: application/x-www-form-urlencoded',
        'content' => $postdata
    )
);
$context  = stream_context_create($opts);

// Smart-in にリクエスト
$response = file_get_contents(SMARTIN_URL, false, $context);

echo $response;
```